CTJ: INPUT-OUTPUT BASED RELATION
COMBINATORIAL TESTING STRATEGY
USING JAYA ALGORITHM

NG YEONG KHANG

BACHELOR OF COMPUTER SCIENCE
(SOFTWARE ENGINEERING)

UNIVERSITI MALAYSIA PAHANG
SUPERVISOR’S DECLARATION

I hereby declare that I have checked this thesis and in my opinion, this thesis is adequate in terms of scope and quality for the award of the degree of Bachelor of Computer Science (Software Engineering).

____________________
(Supervisor’s Signature)
Full Name : Dr. AbdulRahman al-Sewari
Position : Senior Lecturer
Date : 8 January 2019
STUDENT’S DECLARATION

I hereby declare that the work in this thesis is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at Universiti Malaysia Pahang or any other institutions.

________________________________________
(Student’s Signature)
Full Name : Ng Yeong Khang
ID Number : CB15092
Date : 8 January 2019
CTJ: INPUT-OUTPUT BASED RELATION COMBINATORIAL TESTING STRATEGY USING JAYA ALGORITHM

NG YEONG KHANG

Thesis submitted in fulfillment of the requirements for the award of the degree of Bachelor of Computer Science (Software Engineering)

Faculty of Computer Systems and Software Engineering
UNIVERSITI MALAYSIA PAHANG

JANUARY 2019
ACKNOWLEDGEMENTS

First, I would express my deepest appreciation to my respected final year project supervisor, Dr. AbdulRahman al-Sewari in guiding me to complete this thesis which has the title of CTJ: Input-Output Based Relation Combinatorial Testing Strategy Using Jaya Algorithm. He always welcomes me to consult him anytime whenever I faced any problem in doing this final year project. His expertise in software testing gave me insight and well understanding on combinatorial testing and this knowledge accelerates my progress in completing this thesis.

Moreover, I would like to thank to my family members especially my dear parents in supporting me throughout the implementation of my final year project. They always give me advice and backing when I was no idea to solve the trouble encountered. The strong mental support given by them is a great motivation for me to complete this thesis.

Furthermore, a special thanks goes to all friends of mine who gave their helping hands when I need any assistance from them. Their willingness in helping me should be acknowledged as their contributions made the progress of my final year project run uneventful.

Finally, a special gratitude I want to give to all lecturers and staff in Faculty of Computer Systems and Software Engineering, Universiti Malaysia Pahang that helped me directly and indirectly in completing this final year project. Their precious helps made me able to work smoothly throughout this final year project.
ABSTRAK

Pengujian perisian adalah salah satu unsur yang penting dalam pembangunan perisian. Kebanyakan masa, sistem yang diuji mempunyai lebih daripada satu input dan pengujian setiap kombinasi input adalah hampir mustahil kerana masa pelaksanaan kes ujian terlalu panjang. Pengujian kombinatorial ialah satu cara untuk menggantikan ujian menyelesaikan melalui pengujian setiap nilai input dan setiap kombinasi antara parameter. Pengujian kombinatorial boleh dibahagikan kepada tiga jenis iaitu interaksi kekuatan seragam, interaksi kekuatan berubah-ubah dan hubungan berdasarkan input-output (IOR). Pengujian kombinatorial IOR hanya menguji kombinasi penting yang dipilih oleh penguji. Kebanyakan penyelidikan dalam pengujian kombinatorial menggunakan interaksi kekuatan seragam dan berubah-ubah tetapi terdapat hanya beberapa kajian yang menangani IOR. Oleh hal sedemikian, pengujian kombinatorial IOR dipilih untuk dikaji dalam kajian ini. Untuk mengatasi masalah pengoptimalan gabungan, algoritma Jaya dicadangkan untuk digunakan dalam projek ini disebabkan algoritma metaheuristik pantas dalam pengoptimuman dan strategi ini dinamakan sebagai CTJ. Hasil penerapan algoritma Jaya dalam pengujian kombinatorial input-output dapat diterima kerana menghasilkan jumlah kes ujian yang hampir optimum dalam tempoh masa yang memuaskan.
ABSTRACT

Software testing is a vital part in software development lifecycle. Most of the time, system under test has more than one input and testing of every combinations of inputs is almost impossible as the time of execution of test case is outrageously long. Combinatorial testing is the way to encounter exhaustive testing through the testing of every input values and every combination between parameters. Combinatorial testing can be divided into three types which are uniform strength interaction, variable strength interaction and input-output based relation (IOR). IOR combinatorial testing only test for the important combinations that selected by tester. Most of the researches in combinatorial testing applied uniform and variable interaction strength but there are only few studies feature IOR. Thus, IOR combinatorial testing is selected to be studied in this research. To overcome the combinatorial optimization problem, Jaya algorithm is proposed to apply in this project since metaheuristic algorithm is fast in optimization and this strategy is named as CTJ. The result of applying Jaya algorithm in input-output based combinatorial testing is acceptable since it produces nearly optimum number of test cases in the satisfactory time range.
TABLE OF CONTENT

ACKNOWLEDGEMENTS vi
ABSTRAK vii
ABSTRACT viii
TABLE OF CONTENT ix
LIST OF TABLES xii
LIST OF FIGURES xiii
LIST OF ABBREVIATIONS xv

CHAPTER 1 INTRODUCTION 1
  1.1 INTRODUCTION 1
  1.2 PROBLEM STATEMENT 2
  1.3 AIM AND OBJECTIVES 5
  1.4 SCOPE 6
  1.5 THESIS ORGANIZATION 6

CHAPTER 2 LITERATURE REVIEW 7
  2.1 INPUT-OUTPUT BASED RELATION COMBINATORIAL TESTING 7
  2.2 RELATED WORK 12
    2.2.1 Pure Computational Approach 12
    2.2.2 Natural Based Approach 18
  2.3 JAYA ALGORITHM 27

CHAPTER 3 METHODOLOGY 32
  3.1 INTRODUCTION 32
3.2 METHODOLOGY

3.2.1 Phase 1: Literature Review
3.2.2 Phase 2: Design the Solution
3.2.3 Phase 3: Implementation of the Solution
3.2.4 Phase 4: Test and Evaluation
3.2.5 Phase 5: Documentation

3.3 HARDWARE AND SOFTWARE

3.4 GANTT CHART

CHAPTER 4 IMPLEMENTATION, RESULT AND DISCUSSION

4.1 INTRODUCTION

4.2 IMPLEMENTATION OF CTJ

4.2.1 Level 1: Reading of input values
4.2.2 Level 2: Data analysis and data mapping
4.2.3 Level 3: Combinations of input values generation
4.2.4 Level 4: Test case generation based on Jaya algorithm
4.2.5 Level 5: Finalization of test suite generation

4.3 EXPERIMENTAL RESULTS AND DISCUSSION

4.3.1 Parameter tuning of CTJ
4.3.2 Experiments for input-output based relation
4.3.3 Experiments for uniform interaction strength

CHAPTER 5 CONCLUSION

5.1 INTRODUCTION

5.2 RESEARCH CONSTRAINTS

5.3 FUTURE WORKS
LIST OF TABLES

Table 1.1 All possible input values for “Print” section in Notepad++’s Preferences 4
Table 2.1 All input parameters and their input values 8
Table 2.2 List of all possible combinations of input values with t = 2 9
Table 2.3 Test suite generated through pairwise combinatorial testing 9
Table 2.4 List of all possible combinations of input values with IOR 11
Table 2.5 Test suite generated through IOR 12
Table 2.6 Comparison between natural based approaches 18
Table 2.7 Comparison between natural based approaches 27
Table 2.8 Comparison between Jaya algorithm with existing strategies 31
Table 3.1 Result of data mapping using the input values from Table 2.1 35
Table 3.2 60 input-output relationships (R) that utilized in experiments 52
Table 3.3 The size of test suite of existing IOR strategies using configuration IOR (N, 3^{10}, R) 52
Table 3.4 The size of test suite of existing IOR strategies using configuration (N, 2^3 3^3 4^3 5^1, R) 53
Table 3.5 System configuration for uniform interaction strength experiment 54
Table 3.6 The test suite size of existing strategies using uniform interaction strength 54
Table 3.7 List of needed hardware 55
Table 3.8 List of needed software 56
Table 4.1 The mapped values for both input parameters and their corresponding values 60
Table 4.2 Parameter Setting 64
Table 4.3 Test case size and execution time in 30 input-output relationships configuration 65
Table 4.4 Test case size and execution time in CA(N; 3, 6, 6) configuration 65
Table 4.5 Test case size and execution time of IOR (N, 3^{10}, R) configuration in first IOR experiment 68
Table 4.6 Test case size and execution time of IOR (N, 2^3, 3^3, 4^3, 5^1, R) configuration in second IOR experiment 68
Table 4.7 Test case size and execution time of different configurations in uniform interaction strength experiment 69
LIST OF FIGURES

Figure 1.1 Print section of Preferences of Notepad ++ 3
Figure 2.1 Input-output relationship between P, Q, R, S and X, Y, Z of Program P1 11
Figure 2.2 Example to show Greedy algorithm's problem 13
Figure 2.3 Pseudocode of recursive Greedy algorithm (Cormen et al., 2009) 14
Figure 2.4 Pseudocode of test case generation using concept of Density (Z. Wang et al., 2008) 15
Figure 2.5 Overview of AURA strategy (Ong & Zamli, 2011) 16
Figure 2.6 Pseudocode of interaction pair generation algorithm (Ong & Zamli, 2011) 17
Figure 2.7 Pseudocode of test suite generation algorithm (Ong & Zamli, 2011) 17
Figure 2.8 Pseudocode of actual data mapping algorithm (Ong & Zamli, 2011) 18
Figure 2.9 Pseudocode of Ant Colony Optimization (Blum, 2005) 20
Figure 2.10 The illustration on how ants find the shortest path between food source and their nest (Blum, 2005) 20
Figure 2.11 Crossover operation to generate offspring (Elbeltagi, Hegazy, & Grierson, 2005) 22
Figure 2.12 Pseudocode for Genetic Algorithm (Elbeltagi et al., 2005) 22
Figure 2.13 Pseudocode of Harmony Search algorithm (Abdul Rahman, 2012) 24
Figure 2.14 Pseudocode of Particle Swarm Optimization (Poli et al., 2007) 25
Figure 2.15 Pseudocode of Simulated Annealing algorithm (Xambre & Vilarinho, 2003) 26
Figure 2.16 Flowchart of Jaya Algorithm (R Rao, 2016) 29
Figure 3.1 The flowchart of research methodology 33
Figure 3.2 The flowchart of the execution of IOR combinatorial testing based on Jaya algorithm 34
Figure 3.3 The flowchart of test case generation using Jaya algorithm 37
Figure 3.4 GUI for the input of parameters and its values 38
Figure 3.5 GUI of load from file 39
Figure 3.6 GUI of uniform interaction strength 40
Figure 3.7 GUI of input-output based relation 41
Figure 3.8 Pseudocode on reading the parameters and their corresponding values 42
Figure 3.9 Pseudocode of reading all necessary information for test case generation 43
Figure 3.10 Pseudocode for data analysis 43
Figure 3.11 Pseudocode for data mapping 44
Figure 3.12 Pseudocode of combination of input values generation for uniform
strength interaction 46
Figure 3.13 Pseudocode of combination of input values generation for input-output
relationship 47
Figure 3.14 Pseudocode for test case generation using Jaya algorithm (Part 1) 49
Figure 3.15 Pseudocode for test case generation using Jaya algorithm (Part 2) 50
Figure 3.16 Pseudocode of finalization of test case generation 51
Figure 4.1 Example of completed input parameters and its corresponding values at
the Home page of CTJ 58
Figure 4.2 File selection for Load From File option 59
Figure 4.3 The GUI after reading the data from file 59
Figure 4.4 Error message for duplication of value entered 60
Figure 4.5 The complete details that have to be filled in before test case generation
through uniform interaction strength 61
Figure 4.6 The complete details that have to be filled in before test case generation
through input-output relationships 62
Figure 4.7 The test suite generated through ordinary GUI input 63
Figure 4.8 Test suite generated through Load From File 63
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACO</td>
<td>Ant Colony Optimization</td>
</tr>
<tr>
<td>AETG</td>
<td>Automatic Efficient Test Generator</td>
</tr>
<tr>
<td>AGC</td>
<td>Controller for Automatic Generation Control</td>
</tr>
<tr>
<td>AI</td>
<td>Artificial Intelligence</td>
</tr>
<tr>
<td>CA</td>
<td>Covering Array</td>
</tr>
<tr>
<td>EA</td>
<td>Evolutionary Algorithm</td>
</tr>
<tr>
<td>FIFT</td>
<td>Failure-Triggering Fault Interaction</td>
</tr>
<tr>
<td>GA</td>
<td>Genetic Algorithm</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>HM</td>
<td>Harmony Memory</td>
</tr>
<tr>
<td>HMCR</td>
<td>Harmony Memory Considering Rate</td>
</tr>
<tr>
<td>HMS</td>
<td>Harmony Memory Size</td>
</tr>
<tr>
<td>HS</td>
<td>Harmony Search</td>
</tr>
<tr>
<td>IOR</td>
<td>Input-Output Based Relation</td>
</tr>
<tr>
<td>MCA</td>
<td>Mixed Level Covering Array</td>
</tr>
<tr>
<td>PAR</td>
<td>Pitch Adjustment Rate</td>
</tr>
<tr>
<td>PHSS</td>
<td>Pairwise Harmony Search Testing Strategy</td>
</tr>
<tr>
<td>PID</td>
<td>Proportional-Integral-Derivative</td>
</tr>
<tr>
<td>PSO</td>
<td>Particle Swarm Optimization</td>
</tr>
<tr>
<td>PSTG</td>
<td>Particle Swarm Test Generator</td>
</tr>
<tr>
<td>PV-DSTAT-COM</td>
<td>Photovoltaic Fed Distributed Static Compensator</td>
</tr>
<tr>
<td>QAP</td>
<td>Quadratic Assignment Problem</td>
</tr>
<tr>
<td>SA</td>
<td>Simulated Annealing</td>
</tr>
<tr>
<td>SBPWM</td>
<td>Simple Boost Pulse Width Modulation</td>
</tr>
<tr>
<td>SI</td>
<td>Swarm Intelligence</td>
</tr>
<tr>
<td>SUT</td>
<td>System Under Test</td>
</tr>
<tr>
<td>TLBO</td>
<td>Teaching-Learning-Based Optimization</td>
</tr>
</tbody>
</table>
CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

Software testing is an inevitable process in software development lifecycle to find out the software bugs by validating and verifying the application whether it works as expected and meets the business and technical requirements. A recent report from Tricentis, a leading software testing company in Continuous Testing found that there are 606 recorded software failure that happened around the globe which affected over 3.7 billion people and 314 companies as well as $1.7 trillion in lost revenue and 268 years of downtime (Tricentis, 2018). Therefore, a more effective defect detection approach needed to be carried out to increase the coverage of testing.

Combinatorial testing is a black-box testing technique that generate test cases by combining the values of different test object input parameters using combinatorial optimization strategies (De Vries, Vohra, Economics, & Science, 2003). Taking the study from the failure of medical device application, the failure-triggering fault interaction (FTFI) is 68% for single parameter value, 97% of failures triggered by 2 combination values while the percentage of failures caused by 3 and 4 combination values are 99% and 100% respectively (Kuhn, Wallace, & Gallo, 2004). By using combinatorial testing, all input values of the test objects and interactions between each parameter are tested which result in higher detection of interaction failure compared to single parameter testing.

Combinatorial optimization is a process of searching the optimum number of test cases for combinatorial testing. There are many different optimization strategies that are used to generate the test cases for combinatorial testing such as Harmony Search (A. R. A. Alsewari & Zamli, 2012), Genetic Algorithm (Shiba, Tsuchiya, & Kikuno, 2004b),
Ant Colony Algorithm (Shiba et al., 2004b), Simplified Swarm Optimization (Ahmed, Sahib, & Potrus, 2014), Differential Evolution Algorithm (Liang, Guo, Huang, & Jiao, 2014) and so on. Jaya Algorithm is chosen to be applied in this study as this algorithm has been used in lots of optimization problems in other fields.

Combinatorial testing also known as interaction t-way testing where t represents the interaction strength. There are two types of t-way interaction which are uniform strength t-way interaction and variable strength t-way interaction. The interaction between all parameters are uniform in uniform strength t-way interaction while variable strength t-way interaction involves main uniform interaction and sub-uniform interaction. Both type of interactions will generate all possible interactions between each parameter. Often, some of the interactions generated maybe not even be used in the testing. This waste the precious time and effort of the tester to generate those useless interactions. Hence, input-output based relation (IOR) has been introduced in combinatorial optimization to improve the efficiency in finding optimum number of test case as well as given the flexibility in selecting the desired parameter and its interaction (A. R. A. Alsewari & Zamli, 2012).

1.2 PROBLEM STATEMENT

In most of the software application, often there exists one part of system input required to enter a combination of values or choices. The system under test (SUT) is then needed to test for every combination of input parameter to make sure the actual behaviour of the system is same as expected behaviour since the cost of fixing the defect found after software delivered is much higher. Testing of each combination of values is a time and effort wasting job and this leads to exhaustive testing. Exhaustive testing is an impractical software testing technique and usually impossible to achieve in the real testing environment due to budget available and time constraint to execute all combinations of inputs.
Taking the example from the renowned open source code editor, Notepad ++, the “Print” section in the Preferences as shown in Figure 1 is chosen to show the total number of test cases needed to carry out the testing process via exhaustive testing. There are 22 input parameters required to enter in the “Print” section and all possible input values are shown in Table 1.
REFERENCES


international workshop on Software quality assurance: in conjunction with the 6th ESEC/FSE joint meeting, Dubrovnik, Croatia.