



DESIGN AND IMPLEMENTATION OF AN INTERACTIVE TEST CASE GENERATION (ITCG) FOR SEQUENCE-LESS STRATEGY

By

Dalia Sultana
ID: 151-35-1096

This Report Presented in Partial Fulfilment of the Requirements for the Degree
of Bachelor of Science in Software Engineering

**DEPARTMENT OF SOFTWARE ENGINEERING
DAFFODIL INTERNATIONAL UNIVERSITY**

FALL 2018

Copyright © 2018 by Daffodil International University

APPROVAL

This **Thesis** titled “**Design and Implementation of an Interactive Test Case Generation (ITCG) for Sequence-less Strategy**”, submitted by **Dalia Sultana, 151-35-1096** to the department of Software Engineering, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Bachelor of Science in Software Engineering and approval as to its style and contents.

BOARD OF EXAMINERS

Chairman

Professor Dr. Touhid Bhuiyan

Department Head

Department of Software Engineering

Faculty of Science and Information Technology

Daffodil International University

Internal Examiner 1

Dr. Md. Asraf Ali Associate Professor

Department of Software Engineering

Faculty of Science and Information Technology

Daffodil International University

THESIS DECLARATION

I hereby declare that, this Thesis Report has been done under the supervision of Dr. Md. Mostafijur Rahman Assistant Professor, Department of Software Engineering, Faculty of Science and Information Technology, Daffodil International University. I also declare that this report has been submitted elsewhere for award of any degree.

Submitted By

.....

Dalia Sultana

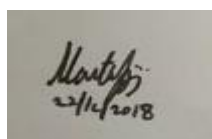
ID: 151-35-1096

Department of Software Engineering

Faculty of Science and Information Technology

Daffodil International University

Certified By



.....

Dr.Md.Mostafijur Rahman

Assistant Professor

Department of Software Engineering

Faculty of Science and Information Technology

Daffodil International University

ACKNOWLEDGEMENT

Alhamdulillah, all praises to the Almighty Allah who gives me the ability, benediction, motivation, Patience and wisdom to complete this research work.

I would like to propagate my gratefulness to my respectful supervisors Dr. Md. Mostafijur Rahman. This Research and Thesis would not departure without him. His excellent guidance, motivation, caring, patience, and providing me with an excellent facilities and environment for doing this research. I am also gratefulness to my parents for their support and pray and care.

TABLE OF CONTENTS

NO	PAGE
APPROVAL	i
THESIS DECLARATION	ii
ACKNOWLEDGEMENT	iii
TABLE OF CONTENTS	iv
LIST OF FIGURES	vi
LIST OF TABLES	vii
LIST OF ABBREVIATIONS	viii
ABSTRACT	ix
CHAPTER 1: INTRODUCTION	
1.1 Background	1
1.1.1 CIIT with Sequence-less Inputs Interaction	1
1.1.2 Sequence-less input interaction with uniform values	1
1.2 Motivation of the Research	3
1.3 Problem Statement	4
1.4 Research Objectives	4
1.5 Research Scope	4
1.6 Research Question	5
1.7 Research Organization	5
CHAPTER 2: LITERATURE REVIEW	
2.1 Nondeterministic and Deterministic Combination Strategies	6
2.2 Existing t-way Strategies for Sequence-less Input Interaction	6
2.2.1 High Level Hyper Heuristic (HHH)	7
2.2.2 Harmony Search Strategy (HSS)	7
2.2.3 Particle Swarm based Test Generator (PSTG)	7
2.2.4 Cuckoo Search Strategy (CSS)	7
2.2.5 Simulated Annealing (SA)	7
2.2.6 Genetic Algorithm (GA)	8
2.2.7 Ant Colony Algorithm (ACA)	8
2.2.8 Bat-Inspired t-way Strategy (BTS)	8
2.2.9 Late Acceptance Hill Climbing (LAHC)	8
2.2.10 Nie Implementation of GA (GA-N)	9
2.3 Summary	9
CHAPTER 3: RESEARCH METHODOLOGY	
3.1 Design of Proposed ITCG Strategy	10
3.2 Design of ITCG strategy for Sequence-less Input Interaction	10
3.2.1 Design of TIDG for sequence-less input interaction	11
3.2.2 Design of t-way Tuple Generator for Sequence-less Input Interaction	11
3.3 Design of Test Case Generator	12
3.4 Summary	15

CHAPTER 4: RESULTS AND DISCUSSION	
4.1 Demonstration of ITCG Strategy Correctness	16
4.1.1 Sequence-less (parameter with uniform values) Interaction	16
4.2 Benchmarking of ITCG Strategy	17
4.2.1 Sequence-less (parameter with uniform values) Interaction	18
4.3 Summary	19
 CHAPTER 5: CONCLUSIONS AND RECOMMENDATIONS	
5.1 Conclusion	20
5.2 Findings and Significant	20
5.3 Recommendations for Future Works	20
5.4 Limitation	21
 REFERENCES	22
APPENDIX A	29
APPENDIX B	30

LIST OF FIGURES

NO	PAGE
1.1Framework of Sequence-less input interaction	1
1.2Fundamental of test processes	4
3.1Framework of ITCG strategy	10
3.2Algorithm for test input data generator for sequence-less input interaction	11
3.3Algorithm for t-way tuple generator	12
3.4 Algorithm of test case generation	14

LIST OF TABLES

NO	PAGE
1.1 Input with uniform values	2
1.2 Exhaustive test cases generated from the Table 2.1	2
1.3 Test cases using t-way strategy 2 for uniform input interaction	3
1.4 3-way tuples for the uniform values	3
4.1 Generated test cases for the system configuration CA (N; 2, 3 ⁴) using ITCG strategy for sequence-less (parameter with uniform values) input interaction	16
4.2 Generated 2-way tuples for the system configuration CA (N; 2, 3 ⁴) using ITCG strategy for sequence-less (parameter with uniform values) input interaction.	17
4.3 Benchmarking of ITCG (uniform values) with existing nondeterministic and deterministic based t-way strategies	18

LIST OF ABBREVIATIONS

ASP	Answer Set Programming
ACTS	Advanced Combinatorial Testing Suite
AI	Artificial Intelligence
ACA	Ant Colony Algorithm
AETG	Automatic Efficient Test Generator
BTS	Bat-Inspired t-way Strategy
CS	Covering Array
CIIT	Combinatorial Input Interaction Testing
CE	Combinatorial Explosion
CP	Control Programing
CV	Condition Value CSS Cuckoo Search Strategy
CSS	Cuckoo Search Strategy
GUI	Graphical User Interface
GTway	Generalized T-way Test Suite Generator
GA	Genetic Algorithm
GA-N	Nie Implementation of GA
HHH	High level Hyper Heuristic
HSS	Harmony Search Strategy
ITCG	Integrated T-way Test Suite Generator
ITCG	Input Test Case Generator
IPOG	In Parameter Order General
LAHC	Late Acceptance Hill Climbing
mAETG	Modified Automatic Efficient Test Generator
PSTG	Particle Swarm Test Generator
SDLC	Software Development Life Cycle
SA	Simulated Annealing
TVG	Test Vector Generator
Tconfig	Test Configuration
TIIT	T-way Input Interaction Testing
TIDG	Test Input Data Generator
TTG	T-way Tuple Generator
TCT	Test Case Generator
TID	Test Input Data

ABSTRACT

Increasing number of input sizes are caused by the exponential growth of test input interaction and create a large input space. The problem examine is needed to do so fast that even the fastest computers require an insufferable amount of time. It limits the ability of computers to solve large input space problems. Only less amount of test case can solve the problem. Since twenty years many useful t-way strategies have been developed to reduce test case size. Deterministic and non-deterministic search strategies are used to design T-way (sequence-less) strategy such as, High level Hyper Heuristic (HHH), Harmony Search Strategy (HSS), Cuckoo Search Strategy (CSS), Particle Swarm Test Generator (PSTG), Simulated Annealing (SA), Genetic Algorithm (GA), Ant Colony Algorithm (ACA), Bat-Inspired t-way Strategy (BTS), Late Acceptance Hill Climbing (LAHC), Nie Implementation of GA (GA-N), Automatic Efficient Test Generator (AETG), Modified Automatic Efficient Test Generator (mAETG), In Parameter Order General (IPOG), Test Vector Generator (TVG), Generalized T-Way Test Suite Generator (GTWay), Density, para order etc. Sequence-less strategy indicates the inputs are taken as parameterized. From the literature it is found that the T-way strategy for sequence-less input interaction is an NP-hard problem. So no one can get optimum solution for every combination of system configuration. In this research an algorithm is proposed and implemented to enhance the T-way input interaction test strategy (sequence-less). To check the effectiveness, the proposed algorithm is compared with the other renown deterministic and non-deterministic search based T-way strategies. The result help to show that the strategy (for sequence-less input interaction) able to generate feasible results and minimize the number of test cases compared with other strategies.

Keyword: T-way testing, Combinatorial input interaction, Deterministic and non-deterministic combinatorial strategy, Deterministic and non-deterministic searching algorithm.

CHAPTER 1

INTRODUCTION

1.1 Background

At present, software systems controlled modern society. The multi-functional and complex designed software has multiple inputs. As a result sufficient software testing is helped while regarding the multiple numbers of input interactions software verification and validation. Insufficient software testing may cause oversight.

1.1.1 CIIT with Sequence-less Inputs Interaction

Fig. 2.1 shows the framework of sequence-less input interaction, where the W, X, Y, Z inputs with two values in each. The sequence-less input interaction can be interaction with uniform values.

1.1.2 Sequence-less input interaction with uniform values

In this interaction, each input holds same number of values. For example, {W, X, Y, Z} is a set of four inputs with two values {w1, w2}, {x1, x2}, {y1, y2} and {z1, z2} (in Table 1.1). Then the exhaustive input interaction is v^p , ie., 2^4 or 24. The Table 2.2 shows the exhaustive test cases (generated from the Table 1.1) are 16. When increasing number of inputs and values effect on cost and time.

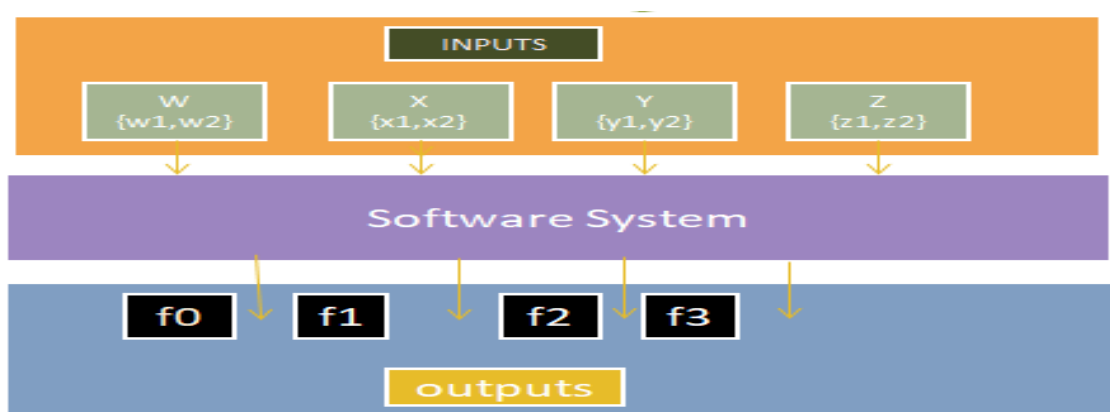


Figure 1.1: Framework of sequence-less input interaction (Othman, 2012)

Table 1.1: Inputs with uniform value

Input	W	X	Y	Z
Values	w1 w2	x1 x2	y1 y2	z1 z2

Table 1.2: Exhaustive test cases generated from the Table 1.1

No.	Exhaustive test case
1	[w1,x1,y1,z1]
2	[w1,x1,y1,z2]
3	[w1,x2,y1,z1]
4	[w1,x2,y1,z2]
5	[w1,x1,y2,z1]
6	[w1,x1,y2,z2]
7	[w1,x2,y2,z1]
8	[w1,x2,y2,z2]
9	[w2,x1,y1,z1]
10	[w2,x1,y1,z2]
11	[w2,x2,y1,z1]
12	[w2,x2,y1,z2]
13	[w2,x1,y2,z1]
14	[w2,x1,y2,z2]
15	[w2,x2,y2,z1]
16	[w2,x2,y2,z2]

The t-way strategy can reduce the number of test cases. In t-way strategy is made based on the specific input strength t, where t is less than the number of input parameter. Consider the test cases are needed to design for a system with four parameters (P) with two values (V) in each (as in Table 1.1). The t-way strategy is used to generate test cases for the given system configuration where t indicates the strengths (2, 3, 4, 5, 6, ..., N). This t-way strategy is designed with Covering Array (CA).

Covering Array (CA): The CA can be defined as CA (N; t, k, v), where N is the number of final test cases, t is strength, the number of independent input parameter is k, and v is the discrete value. The size of the CA can be defined as N x k, where each column represents a parameter and each row represents a test case. The t is any t columns of the array and each of the v^t possible t-way tuples appears at least once (Kuhn, et. al., 2010). For example, suppose a system consists k parameters P_1, P_2, \dots, P_k . Each p_i consists v_i values, where $1 \leq i \leq k$. The number of test candidate can be found from $v_1 \times v_2 \times v_3 \times \dots \times v_k$. We are needed to cover all the t-way tuples but it is too expensive to test all the input combination. Therefore, method is needed to apply and solve some testing criterion.

Table 1.3: Test cases using t-way strategy for the uniform input interaction

No.	Test case
1	[w1,x1,y1,z1]
2	[w1,x2,y2,z1]
3	[w1,x2,y1,z2]
4	[w1,x1,y2,z2]
5	[w2,x2,y1,z1]
6	[w2,x1,y1,z2]
7	[w2,x1,y2,z1]
8	[w2,x2,y2,z2]

Table 1.4: 3-way tuples for the uniform values

No	3-way tuple	No	3-way tuple	No	3-way tuple	No	3-way tuple
1	[w1, x1, y1]	9	[w1, x1, z1]	17	[w1, y1, z1]	25	[x1, y1, z1]
2	[w1, x1, y2]	10	[w1, x1, z2]	18	[w1, y1, z2]	26	[x1, y1, z2]
3	[w1, x2, y1]	11	[w1, x2, z1]	19	[w1, y2, z1]	27	[x1, y2, z1]
4	[w1, x2, y2]	12	[w1, x2, z2]	20	[w1, y2, z2]	28	[x1, y2, z2]
5	[w2, x1, y1]	13	[w2, x1, z1]	21	[w2, y1, z1]	29	[x2, y1, z1]
6	[w2, x1, y2]	14	[w2, x1, z2]	22	[w2, y1, z2]	30	[x2, y1, z2]
7	[w2, x2, y1]	15	[w2, x2, z1]	23	[w2, y2, z1]	31	[x2, y2, z1]
8	[w2, x2, y2]	16	[w2, x2, z2]	24	[w2, y2, z2]	32	[x2, y2, z2]

In the above sample input in the Table 1.1, using t-way strategy, a system configuration can be defined by CA ($N; 3, 2^4$), where the number of test cases is N , the interaction strength is 3, the uniform values is 2 and 4 indicates number of parameters. Table 1.3 shows the generated test cases $N=8$. The 32, 3-way tuples (combination of 3 Values) are shown in Table 1.4. All of the 3-way tuples are covered must be covered by the generated test cases in Table 1.3.

1.2 Motivation of the Research

In combinatorial testing test case has been increased. So, we need to reduce test case by the efficient way. So, this research applies Nondeterministic searching algorithm. Nondeterministic means all time get same output for same input. It is an NP hard problem; no one can challenge to generate minimum number of test cases for every test configuration (Othman, 2012; Afzal et al., 2009). Motivated by the above problems, in this thesis, a new t-way strategy has been proposed.

1.3 Problem Statement

Software test design Combinatorial Explosion (CE) is an important issue. Furthermore, the exhaustive input interaction testing is not fully practiced by the test engineers when time and cost constraints (Williams & Robert, 2001).

Since 1995, there are many sequence-less t-way test strategies are developed. Many Meta – heuristic searching algorithm was use to solve this problem. Like SA(Cohen et al.,2008), FA (Zamli et al.,2018), HHH (Zamli et al., 2016; Zamli et al., 2017) is based on a tabu search Algorithm. Everybody compare their result with another algorithm’s result. As a result they can understand strength and limitations of each strategy, and highlighted the possible research for future work in this area. (Alsewari & Zamli, 2014).

1.4 Research Objective

The aim and objective of this research is to design and evaluate a new strategy based on t-way strategy. To fulfills the aim, the following objectives are taken under consideration:

- To design and implement a t-way sequence-less input interaction test strategy.
- To evaluate the performance of the proposed system with other competing strategies on the basis of the generated test suite size.

1.5 Research Scope

This research focus is to design and develop an integrated t-way strategy supporting sequence-less input interaction to generate test suite (Fig. 1.2). These processes are the media between planning and test execution processes. The test implementation and execution activity involves run the tests. The exit criteria is defined time when test planning and before test execution started. Test closure activities concentrate on making sure that everything is well organized (Morgan et al., 2015).

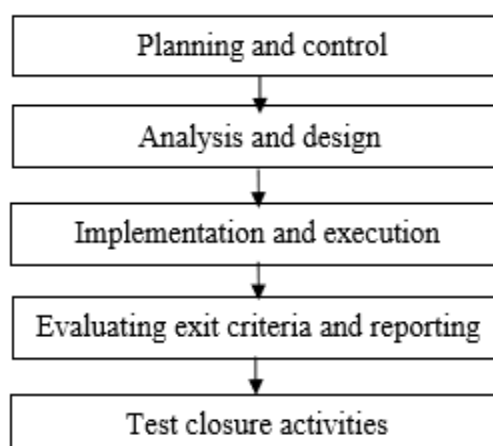


Figure 1.2: Fundamental of test processes (Morgan et al., 2015)

In this research, the design of a new t-way strategy is taken as the main target, which can be used to generate optimum or minimum number of test cases. We also focus time and space complexity can be ignored.

1.6 Research Questions

1. How to design and implement a t-way sequence-less input interaction test strategy?
2. How to evaluate the performance of the proposed system with other competing strategies on the basis of the generated test suite size?

1.7 Thesis Organization

The problem statement, research objective, research scopes, research motivation, as well as research question are introduced in this chapter. The rest of the thesis is organized as follows.

Chapter 2 begins with the definitions and examples of some terminologies underlying of this research. It can do combinatorial methods for t-way test strategies considering sequence-less input interaction.

Chapter 2 begins with the definitions and examples of some terminologies underlying of this research. It can do combinatorial methods for t-way test strategies considering sequence-less input interaction.

Chapter 3 presents the design and development of a new strategy for software input interaction testing based on t-way strategy which considers sequence-less input interaction.

Chapter 4 discusses the evaluation of FSSA for sequence-less input interaction on the basis of finding the correctness of the generated test cases.

Finally, Chapter 5 summarizes the Acquirement and limitation of the proposed design and implementation. Chapter 5 concludes this research work with some recommendation of future direction.

CHAPTER 2

LITERATURE REVIEW

The enacting terminologies related to t-way strategy describe by this chapter. The existing t-way strategies supporting sequence-less input interaction are analyzed. A new algorithm for t-way strategy is analyzed to get the usefulness of assumption.

2.1 Nondeterministic and Deterministic Combination Strategies

The existing t-way strategies can be divided into two groups, like as, nondeterministic and deterministic combination strategy.

The nondeterministic combination strategies, same input parameter model may lead to different test suites. Some existing nondeterministic t-way strategies which adopt heuristic search algorithms are as follows: High level Hyper Heuristic (HHH) (Zamli et al., 2017), Harmony Search Strategy (HSS) (Alsewari & Zamli, 2011), Particle Swarm based Test Generator (PSTG) (Ahmed et al., 2012a; Ahmed et al., 2012b), Cuckoo Search Strategy (CSS) (Nasser et al., 2015), Simulated Annealing (SA) (Cohen et al., 2003), Genetic Algorithm (GA) (Shiba et al., 2004), Ant Colony Algorithm (ACA) (Shiba et al., 2004), Bat-inspired Testing Strategy (BTS) (Alsariera & Zamli, 2015).

The deterministic t-way strategies always generate the same test suite for every execution and give same input parameters, values and strengths. Some deterministic t-way strategies are as follows: In Parameter Order for N-way test (IPO-N) (Nie et al., 2005), Automatic Efficient Test Generator (AETG) (Cohen et al., 1994; Cohen et al., 1997), Modified Automatic Efficient Test Generator (mAETG) (Cohen, 2004), In-Parameter-Order-General (IPOG) (Lie et al., 2007), Modified In-Parameter-Order-General (MIOPG) (Younis et al., 2011), Intelligent Test Case Handler (ITCH) (Hartman et al. 2007), Jenny (Jenkins, 2005), Test Vector Generator (TVG) (Arshem, 2010), Test Configuration (TConfig) (Williams, 2000), Generalized T-Way Test Suite Generator (GTWay) (Zamli et al., 2011), Density (Bryce & Colbourn, 2009).

2.2 Existing t-way Strategies for Sequence-less Input Interaction

In the last decade number of method have been developed for enhancing t-way strategy for sequence-less input interaction. Now I will describe.

2.2.1 High Level Hyper Heuristic (HHH)

The High Level Hyper Heuristic (HHH) (Zamli et al., 2017) strategy is a hybrid t-way test case generation strategy. The four low level meta-heuristic algorithms is selection and acceptance based on the improvement, diversification and intensification operator. The adopted low level meta-heuristic algorithms are designed for continuous problems.

2.2.2 Harmony Search Strategy (HSS)

Alsewari and Zamli (2011), meta-heuristic algorithm adopted harmony search (HS) for t-way test strategy to generate test suite. It is population-based algorithm. The HS uses a probabilistic-gradient in its search space and to select the current solution to adopt mathematical equations for better solution. It is proved that the harmony search algorithm perform well in solving entirely interactive combinatorial problems (Alsewari and Zamli, 2011).

2.2.3 Particle Swarm based Test Generator (PSTG)

Ahmed et al. (2012a; 2012b) designed a t-way test suite generation strategy. It is called particle swarm test generator (PSTG). It is obtained by particle swarm optimization (PSO) (Ahmed et al., 2012a; Ahmed et al., 2012b; Mahmoud & Ahmed, 2015). It is also population based optimization method (Kennedy & Eberhart, 1995a; Kennedy & Eberhart, 1995b). PSO include a group of particles with insignificant mass and volume and which move through hyperspace.

2.2.4 Cuckoo Search Strategy (CSS)

Cuckoo search strategy (CSS) (Nasser et al., 2015) is a recent strategy for t-way test generation. It creates random initial nests. Each egg in a nest represents a vector solution indicates a test case. Firstly, a new nest is created through levy flight path (Yang & Deb, 2009). Then it is appreciated against the existing nests. If there is found a better result, the new nest is replaced as present nest. Secondly, CS has probabilistic elitism in order to maintain elite solutions for the next generation.

2.2.5 Simulated Annealing (SA)

Cohen et al. (Cohen et al., 2003) used simulated annealing (SA) to solve t-way combinatorial problem. This is also a heuristic searching method to acquire optimal test suite. In this technique, first feasible solution is set as a best solution then compare with the best solution. A transformation function is used to select the next feasible solution. Two things are used to control the iteration like cooling rate and temperature (Cohen et al., 2003b).

2.2.6 Genetic Algorithm (GA)

Genetic algorithm (GA) proposed for t-way test strategy to generate test suite (Shiba et al., 2004). It is the process of natural selection. It begins with randomly created test cases, based on chromosomes. These crossover and mutation are happening until a termination criterion is met. The goodness of a candidate function is estimated by using a fitness function. A selection function selects a number of good candidate solutions. The best chromosomes are selected and added to the final test suite.

2.2.7 Ant Colony Algorithm (ACA)

Ant colony optimization (ACO) algorithm adopted on t-way test strategy (Shiba et al., 2004). It is the behavior of natural ant colonies to find paths from the colony to food. The candidate solutions are determined by each path from a starting point to an ending point that is associated with the candidate solution. The amount of pheromone deposited in each ant movement path is selected based on the candidate solution. The next candidate solution is based on the larger number of pheromone. Finally, there may have a possibility to achieve near optimum or optimum solution to the target problem.

2.2.8 Bat-Inspired t-way Strategy (BTS)

Alsariera & Zamli (2015), bat algorithm adapted for t-way strategy to generate test suite, which is called bat-inspired testing strategy (BTS). The bat algorithm (BA) (Yang, 2010) is a natural-inspired algorithm. The interpretation of the nature may not be perfect. The BA is a population optimization algorithm. The bats find their best moving dimension from their position and velocity. In every iteration, the bat algorithm provides an exhaustive local search method throughout its random walk behavior to find the best solution.

2.2.9 Late Acceptance Hill Climbing (LAHC)

Late Acceptance based Hill Climbing (LAHC) is a heuristic search algorithm (Zamli et al., 2015). When a candidate cost function is better (or equal) which accepts non-improving moves. Each current solution is employed during the later (not immediate) acceptance procedure. LAHC is started from a randomly generated initial solution and it evaluates a new candidate in order to accept or reject at each iteration. The last element is compared with the candidate cost of the list and if not worse than accepted. After the acceptance procedure, the cost of the new current solution is inserted into the beginning of the list and the last element is removed from the end of the list. When the inserted current cost is equal to the candidate's cost in the case of accepting only, but in the case of rejecting it is equal to the previous value (Burke & Bykov, 2017).

2.2.10 Nie Implementation of GA (GA-N)

The GA-N is the upgraded version of GA (Shiba et al., 2004). where N indicates N (N=2,3,4,5,6...) number of interaction.

2.4 Summary

In combinatorial input interaction testing the t-way test strategy focus is to reduce number of test cases. Number of researchers are doing research on t-way test strategy for sequence-less input interaction and found that it is NP-hard problem (Shiba et al., 2004; Younis et al., 2010; Othman & Zamli, 2011; Nie & Leung, 20). No researcher can claim that their strategy able to produce the optimum number of test cases for all configuration. Therefore, there is still space for research to design algorithm for t-way test strategy to get lower number of test cases. Next chapter discusses about the proposed t-way strategy algorithm design and implementation.

CHAPTER 3

RESEARCH METHODOLOGY

Sequence-less t-way strategies are NP-hard problem. Now I will discuss a new design on test case tuple test data (ITCG) generate strategy which supports sequence-less input interaction. ITCG strategy for sequence-less input interaction is considered input parameters with uniform values. The overall design is discussed.

3.1 Design of Proposed ITCG Strategy

Fig. 3.1 shows and describes an overview of the proposed ITCG strategy. The design of ITCG strategy consists of three parts, such as, test input data generator (TIDG), tuple generator (TG) and test case generator (TCG). The TCG is designed to generate final test suite. However, the ITCG strategy is designed to support higher number of strength and higher number of input test configuration.

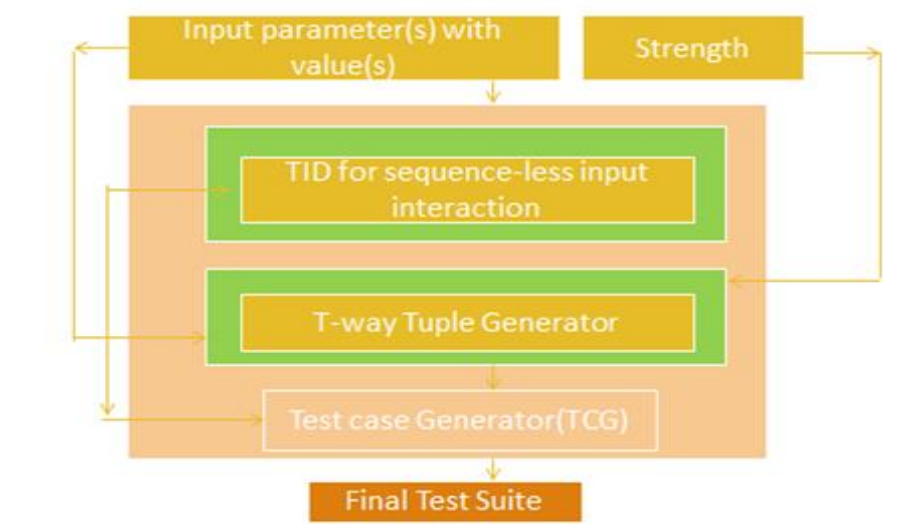


Figure 3.1: Framework of ITCG strategy

3.2 Design of ITCG strategy for Sequence-less Input Interaction

The design of ITCG strategy for sequence-less input interaction consists of test input data and t-way tuple generate then create final test case or test suite. The coverage of t-way tuples check by test input data generator uses TID. A condition value is also used to compare the t-way tuple coverage.

3.2.1 Design of TIDG for sequence-less input interaction

The TIDG consists of TID for sequence-less input interaction. The generated lists of TID are stored into an exhaustive array. The TID are used for test case generator and the t-way tuple generator. The number of TID can be calculated from the number of variables associated with the input parameter.

Algorithm 3.1: Test data generator for sequence-less input interaction

Input: Position(pos)

Outputs: The number of generated input data.

Process:

1. If pos>input **then**
2. **For** i=1 to input **do**
3. Set numeric values of exhaustive test cases into a
4. **End For**
5. **For** i=input+1 to 14
6. Initialize values of Structure a to {0,0}
7. **End For**
8. Store numeric values of a to Exhaustive
9. Call tupleGen () function for current Exhaustive Test Case
10. **End IF**
11. **For** j=1 to values **do**
12. Mark the current position as taken
13. Increment pos by 1 and Go to Step 1
14. **End For**

End of Test Input Data Generator

Figure 3.2: Algorithm for test input data generator.

Like a test configuration CA (N;T, V^P), where N the number of test cases, T strength, V values and P parameters. The total number of TID can be calculated from the P power of V (V^P). Example: configure CA (N; 2, 2^3) where T=2, V=2 and P=4. Test input data is 24 or 16. The input parameters are shown in Table 2.1. The expected lists of TID from Table 2.1 are shown in Table 2.2

3.2.2 Design of t-way Tuple Generator for Sequence-less Input Interaction

Now algorithm 3.2 shows the generating t-way tuples. The lists of TID are used to generate t-way tuples. Loop need to continue to generate the t-way tuples. The number of input parameters and values are needed to generate t-way tuples. For the number of parameters 3, 4, 5, 6, 7, 8, 9, 10 and values are 23-1 or 7, 24-1 or 15, 25-1 or 31, 26-1 or 63, 27-1 or 127, 28-1 or 255, 29-1 or 511, 210-1 or 1023 respectively. The specific number of input parameter generate of t-way tuple.

Algorithm 3.2: t-way tuple generator for sequence-less input interaction**Input:** Input: Position(pos), Counter(cnt), Index(idx)**Outputs:** All tuples for each exhaustive test cases (TempTuple), All unique tuples (Tuple)**Process:**

1. If cnt = ways **then**
2. Initialize paired values of Structure a to {0,0}
3. **For** i=0 to cnt-1 **do**
4. Set numeric values of tuples into a
5. Convert numeric values of a to String tmp
6. **End For**
7. Store String values from tmp to Tuple
8. Store String values of a to Temp Tuple
9. **Return**
10. **End IF**
11. If pos>input **then**
12. **End IF**
13. Mark the current position as taken
14. Increment pos by 1 and cnt by 1 and Go to Step 1
15. Increment pos by 1 and Go to Step 1
16. **Return** T-way Pairs list.

End of Tway Tuple Generator

Figure 3.3: Algorithm for t-way tuple generator

For test configuration CA (N; 3, 2⁴). Algorithm 3.2 (Fig. 3.3) can be generated t-way tuple. Assume the parameters are W, X, Y and Z refer to the Table 2.1 in Chapter 2. The number of 3-way tuple depends on the sum of the 3-way combination of parameters multiplied by their number of values. The 3-way combinations of the four parameters W, X, Y and Z are (W, X, Y), (W, X, Z), (W, Y, Z) and (X, Y, Z). The number of 3-way tuple can be generated (2*2*2) + (2*2*2) + (2*2*2) + (2*2*2) or 32.

3.3 Design of Test Case Generator

Final test suite generation show by the algorithm 3.3. These process complete four phases. First phase, the selection of strategy is made for sequence-less input interaction. Second phase, the corresponding input data generator is called to generate input interaction data. For sequence-less strategy, the test input data generator and tuple generator generated data are stored into two different data set respectively. The data set are used for test case generator. Third phase, a condition value is generated for tuple search. The condition value is depending on t-way tuple coverage of a test candidate.

For sequence-less input interaction, the condition value define by an integer number, which is generated from the maximum number of t-way tuple from each test candidate. The number of input parameters (P) and strength (t) combination is calculated. If the total number of generated t-way tuple and condition value are known when any test configuration the optimum number of test cases can be found. The optimum number of test cases in a test suite is found by taking fist test case and search left and right from this test case and also take a length how much test cash is search from left and right. The optimum number of test cases always found as integer value not found another value. Table A.1 in Appendix A shows optimum number of test cases for some sequence-less (uniform values respectively) input test configurations.

The above requirements for the proposed ITCG strategy for sequence-less input interaction is designed to fulfill the algorithm in Fig. 3.3

The completion of all TID elements checking effect on the condition value to reduce by 1. This algorithm also checks that all the nodes in the t-way tuple are covered or not. The fully covered tuple indicates the search is complete. All nodes in t-way tuple are not covered which indicates the generated test suite is wrong when the condition value decreased to zero. On the other hand, all nodes in t-way tuple tree are covered, indicates the generated test cases are correct when the condition value is greater than zero.

Algorithm 3.3: Test case generator

Input: Pivot (pivot), Jump (jump)

Output: Taken Exhaustive test cases after Searching.

Process:

1. Initialize pivot and jump size
2. **While until** Tuple is empty **do**
3. Mark pivot as taken and store pivot into vector ans
4. For i=0 to TempTuple[pivot],size()-1 **do**
5. **IF** ith Tuple Exists in Tuple **then** delete ith tuple from Tuple
6. **End IF**
7. **End For**
8. **IF** Tuple is empty **then Break**
9. **End IF**
10. Set k=pivot+1,d=0,cv=0,idx=0
11. **While until** k<Exhaustive AND cv<jump **do**
12. **IF** k is marked as taken **Then**
13. Set k=k+1
14. Go to Step 11
15. **End IF**
16. Set cnt=0
17. **For** i=0 to TempTuple[k], size()-1 **do**
18. **IF** ith tuple exists in Tuple **set** cnt=cnt+1
19. **End IF**
20. **End For**
21. **IF** cnt>d **Then**
22. Set d=cnt
23. Set idx=k
24. **End IF**

```

25.   Set k=k+1 and cv=cv+1
26. End While
27. Set k=pivot-1 and cv=0
28. While k>=0 AND cv<Jump do
29.   IF k is marketed as taken Then
30.     Set k=k-1
31.     Go to Step 27
32.   End IF
33.   Repeat Steps 15 to 19
34.   IF cnt=d Then
35.     IF pivot -k<idx -pivot Then
36.       Set idx=k
37.     End IF
38.   Else If cnt>d Then
39.     Set d=cnt and idx=k
40.   End IF
41.   Set k=k-1 and cv=cv+1
42. End While
43. Set pivot=idx
44. End While
45. Print taken Exhaustive test cases after Search.

```

End of Test Case Generation

Figure 3.4: Algorithm of test case generation (continue from previous page)

The first test case always can cover maximum number of tuple. When we search left and right from Initial test case if we found 2 different test cases can cover same maximum number of tuple, we will take close test case from initial test case. Then add the selected TID element in final test case list, if the numbers of any TID element generated number of t-way covered tuple are same as the condition value. The respective covered t-way tuple of the test case and the test case itself are deleted from the original t-way tuple list and TID list respectively for the case of sequence-less input interaction, to reduce the redundant searching. While the condition value becomes zero or the original t-way tuple list become empty, searching process are not work. Every iteration the original t-way tuple list is checked whether all the t-way tuples are covered or not. The selected TID are the final test suites indicate by all t-way covered tuple. In this algorithm, the recursive technique is applied to generate test.

3.4 Summary

Number of research paper have been read to get idea on t-way strategy development. Researchers are embedding different searching algorithms (deterministic and non-deterministic) such as AETG, TVG, Jenny, IPOG, Density, mAETG, IPO-N, and HHH.HSS, PSTG, CSS, SA, GA, ACA, BTS, LAHC,GA-N etc. In this thesis a nondeterministic searching algorithm (proposed in the Figure 3.5) is used to get feasible number of test cases.

CHAPTER 4

EXPERIMENTAL RESULTS

This chapter discusses about the performance of test case tuple test data (ITCG) strategy for sequence-less input interactions. This phase discusses the generated results from ITCG strategy and test cases in terms of cover all generated t-way tuples. Benchmarking against the existing strategies is explored by the performance evaluation of ITCG strategy.

4.1 Demonstration of ITCG Strategy Correctness

The demonstration of ITCG strategy correctness is shown in this phase. The demonstration of correctness is based on sequence-less (parameter with uniform values).

4.1.1 Sequence-less (parameter with uniform values) Interaction

The generated test cases, the number of 2-way tuples covered by each test cases as well as the corresponding 2-way tuples generated from the test cases show table 4.1. The system configuration CA (N; 2, 3⁴) is taken as a test sample, where 2 is the strength and 3⁴ indicates 4 parameters with uniform values 3. Here the 4 parameters values are considered as {w1, w2, w3}, {x1, x2, x3}, {y1, y2, y3} and {z1, z2, z3}. System configuration CA (N; 2, 3⁴) generates total 54 2way tuples. Each test case is able to cover compute maximum number of 2-way tuples by using ITCG strategy for sequence-less input interaction. it is observed that tuples the 9 test cases cover total (6*9) and each of the 9 test cases covered 6 2-way or 54 2-way tuples (shown in Table 4.2). The Table 4.2 demonstrates the test cases 1, 2, 3, 4, 5, 6, 7, 8 and 9 cover the tuples (shown in Table 4.2). The Table 4.2 demonstrates the test cases 1, 2, 3, 4, 5, 6, 7, 8 and 9 cover the Table 4.1: Generated test cases for the system configuration CA (N; 2, 3⁴) using TWIIT strategy for sequence-less (parameter with uniform values) input interaction.

Table 4.1: Generated test cases for the system configuration CA (N; 2, 3⁴) using TWIIT strategy for sequence-less (parameter with uniform values) input interaction.

No.	Generated Test Cases	Cases Covered 2-way Tuples	2-way tuple generated by test case
1.	w1,x1,y1,z1	6	{w1,x1},{w1,y1},{w1,z1},{x1,y1},{x1,z1},{y1,z1}
2.	w1,x2,y2,z2	6	{w1,x2},{w1,y2},{w1,z2},{x2,y2},{x2,z2},{y2,z2}
3.	w1,x3,y3,z3	6	{w1,y3},{w1,y3},{w1,z3},{x3,y3},{x3,z3},{y3,z3}
4.	w2,x1,y2,z3	6	{w2,x1},{w2,y2},{w2,z3},{x1,y2},{x1,z3},{y2,z3}
5.	w2,x2,y3,z1	6	{w2,x2},{w2,y3},{w2,z1},{x2,y3},{x2,z1},{y3,z1}
6.	w2,x3,y1,z2	6	{w2,x3},{w2,y1},{w2,z2},{x3,y1},{x3,z2},{y1,z2}
7.	w3,x1,y3,z2	6	{w3,x1},{w3,y3},{w3,z2},{x1,y3},{x1,z2},{y3,z2}
8.	w3,x2,y1,z3	6	{w3,x2},{w3,y1},{w3,z3},{x2,y1},{x2,z3},{y1,z3}

9.	w3,x3,y2,z1	6	{w3,x3},{w3,y2},{w3,z1},{x3,y2},{x3,z1},{y2,z1}
----	-------------	---	---

Table 4.2: Generated 2-way tuples for the system configuration CA (N; 2, 3⁴) using TWIIT strategy for sequence-less (parameter with uniform values) input interaction.

No	2-way tuples	Covered by test cases	No	2-way tuples	Covered by test cases	No	2-way tuples	Covered by test case
1	w1,x1,	1	19	w1,z1	1	37	x1,z1	1
2	w1,x2	2	20	w1,z2	2	38	x1,z2	7
3	w1,x3	3	21	w1,z3	3	39	x1,z3	4
4	w2,x1	4	22	w2,z1	5	40	x2,z1	5
5	w2,x2	5	23	w2,z2	6	41	x2,z2	2
6	w2,x3	6	24	w2,z3	4	42	x2,z3	8
7	w3,x1	7	25	w3,z1	9	43	x3,z1	9
8	w3,x2	8	26	w3,z2	7	44	x3,z2	6
9	w3,x3	9	27	w3,z3	8	45	x3,z3	3
10	w1,y1	1	28	x1,y1	1	46	y1,z1	1
11	w1,y2	2	29	x1,y2	4	47	y1,z2	6
12	w1,y3	3	30	x1,y3	7	48	y1,z3	8
13	w2,y1	6	31	x2,y1	8	49	y2,z1	9
14	w2,y2	4	32	x2,y2	2	50	y2,z2	2
15	w2,y3	5	33	x2,y3	5	51	y2,z3	4
16	w3,y1	8	34	x3,y1	6	52	y3,z1	5
17	w3,y2	9	35	x3,y2	9	53	y3,z2	7
18	w3,y3	7	36	x3,y3	3	54	y3,z3	3

4.2 Benchmarking of ITCG Strategy

This is a nondeterministic based strategy. The reported test suites size is produced from 20 times execution of the program. The ITCG is compared with existing strategies. Test suite sizes define which strategy is best. Other t-way strategies (Zamli, et. al., 2017; Zamli, et. al., 2016; Stardom, 2001; Ahmed, et. al., 2015; Ahmed, et. al., 2012; Cohen, 2005; Garvin, et. al., 2010; Bryce & Colbourn, 2007; Ahmed, et. al., 2012; Alsewari, 2012; Ahmed, 2012) are taken for benchmarking from Only the published test configurations. The comparison show that nondeterministic and deterministic search based t-way strategies. Test suites size indicate by “bold” the minimum size of test suite and others are feasible solution for the configuration of interests. The results are not available in any publication for the particular method which define “-” (dash) sign. The test suite size is optimum and best solution indicates “*” (star) sign.

4.2.1 Sequence-less (parameter with uniform values) Interaction

The results are evaluated by benchmarking with existing t-way strategies for sequence-less (parameter with uniform values) interaction. The nondeterministic based t-way strategies are: HHH (Zamli, et. al., 2016;), HSS (Alsewari, & Zamli, 2012), PSTG (Ahmed, et.al., 2012), CSS (Ahmed, et. al., 2015), SA (Cohen, et. al., 2003), GA (Shiba, et. al., 2004), ACA (Shiba, et. al., 2004), BTS(Alsariera & Zamli, 2015), LAHC (Zamli, et. al., 2015), and GA-N (Shiba, et. al., 2004). Besides the deterministic based t-way strategies are: IPO-N (Nie, et. al., 2005), AETG(Cohen, et. al., 1997), mAETG (Cohen, 2004), IPOG (Lie, et. al., 2007a), MIOGP (Younis, et. al., 2011), ITCH (Hartman, et. al. 2007), Jenny (Jenkins, 2010), TVG(Arshem, 2010), TConfig (Williams, 2000), GTWay(), Density(Wang et al., 2008), ParaOrder (Wang et al., 2007), PICT(Czerwonka, 2006) and ITTSG(Othman, 2012). SA and GA generate better result from most other strategies for lower interaction ($t < 3$) refer to Table 4.7,. For the HHH, CSS, ACA, GA-N, IPO-N, mAETG, AETG strategies, there are no published result found for the system configurations (in Table 4.3) with interaction greater than 3 ($t > 3$).

Table 4.3 Benchmarking of ITCG strategy (uniform values) with existing deterministic and Nondeterministic based t-way strategies

System configuration	Nondeterministic based strategies											
	ITCG	TWITT	HHH	HSS	PSTG	CSS	SA	GA	ACA	BTS	LAHC	GA-N
CA (N; 2, 3 ⁴)	9*	9*	9*	9*	9*	9*	9*	9*	9*	-	-	-
CA (N; 2, 2 ¹³)	9*	19	-	-	17	-	16	17	17	-	-	-
CA (N; 3, 3 ⁴)	33	27	27	27	27	27	-	-	-	-	-	-
CA (N; 3, 3 ⁶)	15	34	33	39	42	43	33	33	33	-	-	52
CA (N; 3, 4 ⁶)	64*	64*	70	-	102	105	64*	64*	64*	-	-	85
CA (N; 4, 5 ⁵)	749	727	-	-	783	-	-	-	-	-	-	-
CA (N; 4, 3 ⁶)	135	132	-	134	-	-	-	-	-	132	132	-
CA (N; 4, 2 ¹⁰)	9*	42	-	-	-	-	-	-	-	-	-	-
CA (N; 5, 2 ⁶)	32*	32*	-	-	-	-	-	-	-	-	-	-
CA (N; 5, 2 ⁸)	64*	64*	-	66	65	-	-	-	-	64*	64*	-
CA (N; 6, 2 ⁷)	64*	64*	-	64*	67	-	-	-	-	64*	64*	-
CA (N; 6, 3 ⁷)	848	848	-	-	-	-	-	-	-	-	-	-
Note: Bold value, ‘*’ and ‘-’ indicate the best, minimum & optimum size of test suite, and no published result found, respectively												

System configuration	Deterministic based strategies											
	ITCG	TWILT	IPO-N	maETG	IPOG	Jenny	TVG	Tconfig	GTWay	Density	Para Order	PICt
CA (N; 2, 3 ⁴)	9*	9*	-	9*	-	-	-	-	-	-	-	-
CA (N; 2, 2 ¹³)	9*	19	-	17	-	-	-	-	-	-	-	-
CA (N; 3, 3 ⁴)	33	27	-	-	39	34	32	32	-	-	-	34
CA (N; 3, 3 ⁶)	15	34	47	38	-	51	48	48	-	63	53	48
CA (N; 3, 4 ⁶)	64*	64*	64*	77	-	112	120	64*	-	64*	106	111
CA (N; 4, 5 ⁵)	749	727	-	-	784	837	849	773	771	-	730	-
CA (N; 4, 3 ⁶)	135	132	-	-	-	140	-	141	-	-	-	142
CA (N; 4, 2 ¹⁰)	9*	42	-	-	46	39	40	45	46	-	45	-
CA (N; 5, 2 ⁶)	32*	32*	-	-	-	-	-	-	-	-	-	-
CA (N; 5, 2 ⁸)	64*	64*	-	-	-	74	-	70	-	-	-	64*
CA (N; 6, 2 ⁷)	64*	64*	-	-	-	87	-	64*		-	-	72
CA (N; 6, 3 ⁷)	848	848	-	-	-	-	-	-	-	-	-	-
Note: Bold value, '**'and '- 'indicate the best, minimum & optimum size of test suite, and no published result found, respectively												

4.3 Discussion

In this research, the execution time is not main issue for the comparison because most implementations are not to be executed on the computer as used in this research. The exact time to generate final test cases cannot ensure nondeterministic search based strategy (Crepinsek, et al. 2014a; Crepinsek, et al. 2014b; Mernik, et al., 2015; Draa, 2015). My algorithm is also gives feasible solution for most of the test input configuration. Three result is best that any kind of previous algorithm is not give minimum solution than my algorithm. Another solution is also best like existing algorithm. This research successfully produces better results in many scenarios that is shown in the Table 4.3.

CHAPTER 5

CONCLUSION AND FUTURE WORKS

In Chapter 4, the experimental results of ITCG strategy are practiced the goals and highlighted the contributions achieved from this research in the field of software test engineering.

5.1 Conclusion

The aim and objectives of this research have been successfully achieved. The findings are formed a guideline for designing input interaction strategy based on uniform strength. In order to achieve the objective detail discussion on the proposed ITCG strategy design modules are done. A new t-way strategy, has been successfully developed to generate feasible and competitive test cases. The developed ITCG strategy supports sequence-less input interaction. The sequence-less input interaction supports uniform input interaction. Finally, the generated test cases are proved based on the covered t-way tuples. The performance evaluation of ITCG is performed by comparing with the existing strategies on the basis of the number of generated test cases.

The ITCG strategy can be an efficient for software assurance that empirical experience shows. The ITCG strategy is designed and developed to support for higher strength and higher numbers of input interactions. This research focuses on the traditional software interaction testing. I think that there are many emerging applications can be tested using the ITCG as similar to existing t-way strategies. The design of ITCG strategy can be extended beyond the application of software testing.

5.2 Findings and Significant

The ITCG strategy, for sequence-less input interaction, the integration of uniform input interaction can be a performance. By the development for sequence-less input interaction, random and input jump techniques are used to cover all t-way tuple efficiently. The significance of this research is this is a new approach to deal with sequence-less CIIT that produces better results in some scenarios that is discussed in Table 4.3.

5.3 Recommendations for Future Works

In this research, a new ITCG strategy is designed and developed carefully to support sequence-less input interactions. ITCG strategy can be made efficient by using many ways. By adopting other random algorithms (such as nondeterministic algorithms) it can be made more efficient. Implementation wise ITCG can be extended in several ways. However, the jump technique is ease for the ITCG strategy to implementation the concurrent and can be reduced the t-way tuple search time. ITCG strategy can be extended for input-output based relations and variable strength interaction for generating test cases. Finally, to establish a more efficient t-way input

interaction test strategy for sequence-less input interaction still deserves for future research.

5.4 Limitation

My research work still has some limitations. In the future, first we should find potential optimal combination of design parameter then apply our approach to more inputs to demonstrate the effectiveness. Second, we want to apply our algorithms to support the software testing work in real software enterprises and verify their effectiveness in software faults detection. My algorithm can be revised and improvised to work with bigger and non-uniform values.

REFERENCES

Ahmed, B. S., Zamli, K. Z., & Lim, C. P. (2012a).

Application of particle swarm optimization to uniform and variable strength covering array construction. *Applied soft computing*, 12(4), 1330-1347. doi:10.1016/j.asoc.2011.11.029.

Afzal, W., Torkar, R. & Feldt, R. (2009).

A systematic review of search-based testing for non-functional system properties. *Information and software technology*, 51(6), 957-976.

Ahmed, B. S., Zamli, K. Z. and Lim, C. P. (2012b).

Constructing a t-way interaction test suite using the particle swarm optimization approach. *International journal of innovative computing, information and control (ICIC)*, 8(1(A)), 431-451.

Ahmed, B. S., & Zamli, K. Z. (2010).

T-way test data generation strategy based on particle swarm optimization. In *proceedings of 2nd international conference on computer research and development*, 93-97. doi:10.1109/iccrd.2010.56.

Ahmed, B. S., Abdulsamad, T. S. & Potrus, M. Y. (2015).

Achievement of minimized combinatorial test suite for configuration-aware software functional testing using the cuckoo search algorithm. *Information and software technology*, 66, 13-29. doi:10.1016/j.infsof.2015.05.005

Alsewari, A. A., & Zamli, K. Z. (2011).

Interaction test data generation using harmony search algorithm. In *proceedings of the IEEE symposium on industrial electronics and applications*, 559-564. doi:10.1109/isiea.2011.6108775

Alsariera, Y. A., & Zamli, K. Z. (2015).

A bat-inspired strategy for t-way interaction testing. *Journal of advanced science letters*, 21(8), 2281-2284. doi:10.1166/asl.2015.6316.

- Arshem, J. (2004).
Test vector generator. Retrieved on April 5, 2017 from
<http://sourceforge.net/projects/tvg>.
- Alsewari, A. A., & Zamli, K. Z. (2014).
An orchestrated survey on t-way test case generation strategies based on optimization algorithms. In proceedings of the 8th international conference on robotic, vision, signal processing & power applications, 255-263. doi:10.1007/978-981-4585-42-2_30.
- Bryce, R. C., & Colbourn, C. J. (2009).
A density-based greedy algorithm for higher strength covering arrays. Software testing, verification and reliability, 19(1), 3753. doi:10.1002/stvr.393.
- Burke, E. K., & Bykov, Y. (2017).
The late acceptance hill-climbing heuristic. European journal of operational research, 258(1), 70-78. doi:10.1016/j.ejor.2016.07.012.
- Cohen, D., Dalal, S., Fredman, M., & Patton, g. (1997).
The AETG system: an approach to testing based on combinatorial design. IEEE transactions on software engineering, 23(7), 437-444. doi:10.1109/32.605761.
- Cohen, D., Dalal, S., Kajla, A., & Patton, G. (1994).
The Automatic Efficient Test Generator (AETG) system. In proceedings of the IEEE international symposium on software reliability engineering, 303-309. doi:10.1109/issre.1994.341392.
- Cohen, M. B. (2004).
Designing test suites for software interaction testing. University of Auckland.
- Cohen, M. B., Colbourn, C. J., & Ling, A. C. (2003a).
Augmenting simulated annealing to build interaction test suites. In proceedings of the 14th international symposium on software reliability engineering (ISSRE'03), 394-405. doi:10.1109/issre.2003.1251061.

Cohen, M., Gibbons, P., Mugridge, W., & Colbourn, C. (2003b).

Constructing test suites for interaction testing. In proceedings of the 25th international conference on software engineering, 38-48. doi:10.1109/icse.2003.1201186.

Cohen, M. B., Colbourn, C.J. & Ling, A. C. H. (2008).

Constructing strength three covering arrays with augmented annealing. Discrete Math, 308, 2709–2722.

Črepinšek, M., Liu, S., Mernik, L., & Mernik, M. (2014a).

Is a comparison of results meaningful from the inexact replications of computational experiments? Soft computing, 20(1), 223-235. doi:10.1007/s00500-014-1493-4.

Črepinšek, M., Liu, S-H., & Mernik, L. (2014b).

Replication and comparison of computational experiments in applied evolutionary computing: common pitfalls and guidelines to avoid them. Applied soft computing, 19, 161-170. doi:10.1016/j.asoc.2014.02.009.

D. Yazdani, S. Sadeghi-Ivrigh, D. Yazdani, A. Sepas-Moghaddam and M. R. Meybodi,

Fish Swarm Search Algorithm: A New Algorithm for Global Optimization, International Journal of Artificial Intelligence, vol. 13, no. 2, pp. 17-45, 2015.

Draa, A. (2015).

On the performances of the flower pollination algorithm - qualitative and quantitative analyses. Applied soft computing, 34, 349-371. doi:10.1016/j.asoc.2015.05.015.

Hartman, A., Klinger T., Raskin L. (2005).

IBM intelligent test case handler. Retrieved on April 5, 2017 from <http://ibm-intelligent-test-case-handler.updatestar.com/en>.

Harrold, M. J. (2000).

Testing: A roadmap. In proceedings of the conference on the future of software engineering, 61-72.

Jenkins, B. (2005).

Jenny test tool. Retrieved on April 5, 2017 from <http://www.burtleburtle.net/bob/math/jenny.html>.

Kennedy, J. & Eberhart, R.. (1995a).

Particle swarm optimization. In proceedings of the IEEE international conference on neural networks, 1942-1948. doi: 10.1109/ICNN.1995.488968.

Kennedy, J. and Eberhart, R. (1995b).

A new optimizer using particle swarm theory. In proceedings of the 6th international symposium on micro machine and human science, 39-43. dio: 10.1109/MHS.1995.494215.

Kuhn, D. R., Kacker, R. N., Lei, Y. (2010).

Practical combinational testing. U.S.. department of commerce, national institute of standards and technology (NIST), Special publication 800-142.

Lei, Y., Kacker, R., & Kuhn, D. R. (2007a).

IPOG: A general strategy for t-way software testing. In proceedings of the 14th annual IEEE international conference and workshops on the engineering of computer-based systems (ECBS'07), 549-556. doi:10.1109/ECBS.2007.47.

Lei, Y., Kacker, R., Kuhn, R., Okun, V., & Lawrence, J. (2007b).

IPOG/IPOGD: Efficient test generation for multi-way combinatorial testing. Journal of software testing, verification and reliability, 18(3), 125-148.

Mernik, M., Liu, S., Karaboga, D., & Črepinšek, M. (2015).

On clarifying misconceptions when comparing variants of the artificial bee colony algorithm by offering a new implementation. Information Sciences, 291, 115-127. doi:10.1016/j.ins.2014.08.040.

Mahmud, T. & Ahmed, B. S. (2015).

An effective strategy for covering array construction with fuzzy logic-based adaptive swarm optimization for software functional testing Use. Expert system with application, 42, 8753-876.

Morgan, P., Hambling, B., Thompson, G., Samaroo, A., Williams, P. (2015).

Software testing: an istqb-bcs certified tester foundation guide. BCS learning & development limited, USA. ISBN 1780172990, 9781780172996.

Nasser, A. B., Alsewari, A. A., & Zamli, K. Z. (2015).

Tuning of Cuckoo Search based Strategy for T-way Testing. ARPN Journal of Engineering and Applied Sciences, 10(19), 8948-8953.

Nie, C., Xu, B., Shi, L., & Dong, G. (2005).

Automatic Test Generation for N-Way Combinatorial Testing. Lecture Notes in Computer Science Quality of Software Architectures and Software Quality, 203-211. doi:10.1007/11558569_15.

Othman, R. R. (2012).

Design of a T-way Test Suite Generation Strategy Supporting Flexible Interactions. Universiti Sains Malaysia (USM).

Othman, R. R. and Zamli, K.Z., (2011).

T-way strategies and its applications for combinatorial testing. International journal on new computer architectures and their applications (IJNCAA), 1(2), 459-473.

Rahman, M. (2017).

Design of a New T-way Strategy for Test Case Generation Supporting Sequence-less and Sequence Input Interaction. PhD thesis, Universiti Malaysia Perlis (UNIMAP).

Stardom, J. (2001).

Metaheuristic and the search for covering and packing array. Simon Fraser University.

Shiba, T., Tsuchiya, T. and Kikuno T. (2004).

Using artificial life techniques to generate test cases for combinatorial testing. In proceedings of the 28th annual international computer software and applications Conference, 01, 72-77.

Tassey, G. (2002).

The economic impacts of inadequate infrastructure for software testing. National institute of standards and technology, RTI Project Number 7007.011.

Williams, A. W. (2000).

Determination of test configurations for pair-wise interaction coverage. In proceedings of the advances in information and communication technology testing of communicating systems, 59-74. doi:10.1007/978-0-38735516-0_4.

Williams, A. W. & Probert, R. L. (2001).

A measure for component interaction test coverage. In proceedings of the International conference on computer systems and applications (AICCSA 2001), 304-311.

Chen, X. Gu, Q. Li, A. and Chen D (2009).

Variable Strength Interaction Testing with an Ant Colony System Approach. 1530-1362/09 \$26.00 © 2009 IEEE DOI 10.1109/APSEC.2009.18

Yang, X., & Deb, S. (2009).

Cuckoo search via lévy flights. In proceedings of the world congress on nature & biologically inspired computing (NaBIC), 210-214. doi:10.1109/nabic.2009.5393690.

Younis, M. I., & Zamli, K. Z. (2011).

MIPOG-an efficient t-way minimization strategy for combinatorial testing. International journal of computer theory and engineering, 3(3), 388-397.

Zamli, K. Z., Din, F., Kendall, G., & Ahmed, B. S. (2017).

An experimental study of hyper-heuristic selection and acceptance mechanism for combinatorial t-way test suite generation. Information sciences, 399, 121-153. doi:10.1016/j.ins.2017.03.007.

Zamli, K. Z., Alkazemi, B. Y., & Kendall, G. (2016).

A Tabu Search hyper-heuristic strategy for t-way test suite generation. Applied soft computing, 44, 57-74. doi:10.1016/j.asoc.2016.03.021.

Zamli, K. Z., Klaib, M. F., Younis, M. I., Isa, N. A., & Abdullah, R. (2011).

Design and implementation of a t-way test data generation strategy with automated execution tool support. *Information Sciences*, 181(9), 1741-1758.
doi:10.1016/j.ins.2011.01.002.

Ziyuan, W., Changhai, N., & Baowen, X. (2007).

Generating combinatorial test suite for interaction relationship. In proceedings of the 4th international workshop on software quality assurance in conjunction, 55-61.
doi:10.1145/1295074.1295085.

APPENDIX A

Table A.1: Theoretically calculated optimum test suite size for sequence-less input interaction (uniform values)

No.	Configuration	No. of Generated T-way Pairs	Condition Value	Optimum number of Test Case
1	CA (N; 2, 3 ⁴)	54	6	9
2	CA (N; 2, 2 ¹³)	312	52	6
3	CA (N; 3, 3 ⁶)	540	15	36
4	CA (N; 3, 4 ⁶)	1280	16	80
5	CA (N; 3, 5 ⁷)	4375	25	175
6	CA (N; 3, 5 ⁶)	2500	20	125
7	CA (N; 4, 5 ⁵)	3125	25	125
8	CA (N; 4, 5 ⁶)	9375	15	625
9	CA (N; 4, 2 ¹⁰)	3360	105	32
10	CA (N; 5, 2 ⁶)	192	12	16
11	CA (N; 5, 2 ⁸)	1792	64	28
12	CA (N; 6, 2 ⁷)	448	14	32
13	CA (N; 6, 3 ⁷)	5103	7	729
14	CA (N; 4, 5 ⁷)	21875	25	875
15	CA (N; 3, 6 ⁶)	4320	30	144

APPENDIX B

Table B.1 shows the summary of supported interaction by the existing t-way strategies support sequence-less or sequence input interaction.

Existing t-way strategies	Sequence-less Input Interaction	
	Uniform Values	Non-uniform Values
Elitist-Flower Pollination Algorithm(FPA)(Zamli et al., 2018)	/	/
High level Hyper Heuristic (HHH) (Zamli et al., 2017)	/	/
Flower Pollination Algorithm(FPA)(Zamli et al., 2015)	/	/
Particle Swarm based Test Generator (PSTG) (Ahmed et al., 2012a; Ahmed et al., 2012b)	/	/
Cuckoo Search Strategy (CSS) (Nasser et al., 2015)	/	/
Simulated Annealing (SA) (Cohen et al., 2003)	/	/
Genetic Algorithm (GA) (Shiba et al., 2004)	/	/
Ant Colony Algorithm (ACA) (Shiba et al., 2004)	/	/
Bat-inspired Testing Strategy (BTS) (Alsariera & Zamli, 2015)	/	/
Late Acceptance based Hill Climbing (LAHC) (Zamli et al., 2015)	/	/
GA-N (Shiba et al., 2004)	/	/
In Parameter Order for N-way test (IPO-N) (Nie et al., 2005)	/	/
Automatic Efficient Test Generator (AETG) (Cohen et al., 1994; Cohen et al., 1997)	/	/
Modified Automatic Efficient Test Generator (mAETG) (Cohen, 2004)	/	/
In-Parameter-Order-General (IPOG) (Lie et al., 2007)	/	/
Modified In-Parameter-Order-General (MIOPG) (Younis et al., 2011)	/	/
Intelligent Test Case Handler (ITCH) (Hartman et al. 2007)	/	/
Jenny (Jenkins, 2005)	/	/
Test Vector Generator (TVG) (Arshem, 2010)	/	/
Test Configuration (TConfig) (Williams, 2000)	/	/
Generalized T-Way Test Suite Generator (GTWay) (Zamli et al., 2011)	/	/
Density (Bryce & Colbourn, 2009)	/	/
Parameter Ordered (ParaOrder) (Ziyuan et al., 2007)	/	/
Pairwise Independent Combinatorial Testing (PICT) (Czerwonka, 2006)	/	/
Integrated t-Way Test Suite Generator (ITTSG) (Othman, 2012)	/	/