PSAT: A PAIRWISE TEST DATA GENERATION TOOL BASED ON SIMULATED ANNEALING ALGORITHM

GOH GHEE HAU

BACHELOR DEGREE OF COMPUTER SCIENCE
(SOFTWARE ENGINEERING)
UNIVERSITI MALAYSIA PAHANG

DECLARATION OF THESIS AND COPYRIGHT

Author's full name : GOH GHEE HAU

Date of birth : 07 JUNE 1992

Title : PSAT: A PAIRWISE TEST DATA GENERATION TOOL BASED ON SIMULATED ANNEALING ALGORITHM

Academic Session : SEMESTER 1 2015/2016

I declare that this thesis is classified as:

CONFIDENTIAL	(Contains confidential information under the Official Secret Act 1972)*
RESTRICTED	(Contains restricted information as specified by the organization where research was done)*
OPEN ACCESS	I agree that my thesis to be published as online open access (Full text)

I acknowledge that University Malaysia Pahang reserve the right as follows:

1. The Thesis is the Property of University Malaysia Pahang.
2. The Library of University Malaysia Pahang has the right to make copies for the purpose of research only.
3. The Library has the right to make copies of the thesis for academic exchange.

Certified By:

(Student's Signature)	(Signature of Supervisor)
New IC / Passport Number	Name of Supervisor
Date:	Date:

DECLARATION

I hereby declare that this research entitled “PSAT: A Pairwise Test Data Generation Tool Based on Simulated Annealing Algorithm” is the result of my own research except the citation in the references. The result has not been accepted for any degree and is not concurrently submitted in candidature of any degree.

Date: 9th December 2015 GOH GHEE HAU

CB12079

SUPERVISOR DECLARATION

I hereby declare that I have read this research and my option in this research is sufficient in terms of scope and quality for the award of the degree of Bachelor of Computer Science (Software Engineering).

Signature	: ….………………………………
Supervisor Name	: Dr. AbdulRahman A. Al-Sewari
Date	: ………………………………….

ACKNOWLEDGEMENTS

Firstly, I would like to express my thanks to my beloved parents who born me and provides opportunity for me to enter local university. I also like to thanks to University Malaysia Pahang (UMP) for giving me a chance and place to learn knowledge and done my bachelor degree.

Besides that, I am very appreciating to have Dr. AbdulRahman A. Al-Sewari as my supervisor for my Final Year Project (FYP) in 2015. I would like to thanks for his advices, guidelines and support for me to complete my research during the whole FYP.

Hereby, I wish to thanks all the lecturers in UMP for passing their knowledge to me to complete my research and studies in UMP. Next, I would like to thank to all my fellow friends and seniors that always support me during the life in UMP.

Last but not least, I would like to thank the UMP RDU130366 Short Term Grants: Development of a Pairwise Testing Tool with Constraint and Seeding Support Based on and Optimization Algorithm, UMP RDU150369 A new Hybrid Variable Interaction Strength Test Data Generation Strategy Based on Harmony Search Algorithm and Cuckoo Search Algorithm, and FRGS RDU130199 Grant: Input Output Relations Harmony Search T-way Testing Strategy, whose partially funded this dissertation.

ABSTRACT

In this information technology era, there is a huge influence of high technology and artificial intelligence when creating new software products in the whole world. To bring high quality software products to the end user, software testing plays important roles and need to be considered during the development stage. Therefore, there is impossible to cover all the test case of the software products and may lead to exhaustive testing. This research is about the research on developing a Pairwise Test Data Generation Tool based on Simulated Annealing (SA) algorithm which named as PSAT. PSAT is used to generate the sufficient test case to reduce the financial resources and time. In PSAT, we using pairwise testing techniques which each interaction of test case considers of two input parameters. Each parameter will have different parameter values that entering by users. In development of PSAT, a prototype of Graphics User Interface (GUI) will be designed and create for user to enter the number of parameters and number of values for each parameter. In this research, the test case will be generated based on the SA algorithm to show that the test case can be generated with sufficient.

ABSTRAK

Dalam era penuh dengan teknologi maklumat ini, terdapat pengaruh besar terhadap teknologi tinggi dan kecerdasan buatan semasa membuat produk perisian baru di seluruh dunia. Untuk membawa produk perisian yang berkualiti tinggi kepada pengguna akhir, ujian perisian memainkan peranan yang penting dan perlu dipertimbangkan semasa peringkat pembanguanan. Oleh sebab itu, adalah mustahil untuk merangkumi semua kes ujian terhadap produk perisian dan ini boleh menyebabkan ujian lengkap terhadap sesuatu produk. Kajian ini adalah mengenai pembangunan terhadap satu *Pairwise Test Data Generation Tool* berdasarkan *Simulated Annealing (SA)* algoritma dengan beri nama *PSAT*. *PSAT* digunakan untuk menjana kes ujian supaya dapat mengurangkan kegunaan sumber kewangan and masa. Dalam *PSAT*, kita menggunakan teknik ujian dari segi pasangan di mana setiap interaksi kes ujian mempunyai dua *parameter input*. Dalam setiap *parameter* akan mempunyai nilai-nilai yang berbeza yang akan dimasukkan oleh pengguna. Dalam pembangunan *PSAT*, satu prototaip *Graphics User Interface (GUI)* akan direka bentuk dan membolehkan pengguna memasukkan nombor *parameter* dan niali bagi setiap *parameter*. Dalam kajian ini, kes ujian dijana mengikut *SA* algoritma untuk menujukan kes ujian dapat dijana dengan mencukupi.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ACA	Ant Colony Algorithm
AETG	Automatic Efficient Test Generator
BA_PTC	A Bat-Inspired Strategy for Pairwise Testing With Constraints Support
GA	Genetic Algorithm
GUI	Graphics User Interface
HS	Harmony Search Algorithm
IPO	In-Parameter-Order
IPOG	In-Parameter-Order-General
IPOG-D	In-Parameter-Order-General with D-construction
IPOG-F	In-Parameter-Order-General with FireEye
IRPS	Intersection Residual Pair Set
JDK	Java Development Kit
LAHC	Late Acceptance Hill Climbing Algorithm
OPAT	One Parameter at a Time
OTAT	One Test at a Time
OS	Operating System
PICT	Pairwise Independent Combinatorial Testing
PSAT	Pairwise Test Data Generation Tool Based on Simulated Annealing Algorithm
RAD	Rapid Application Development
RAM	Random Access Memory

R&D	Research and Development
SA	Simulated Annealing Algorithm
SDLC	Software Development Life Cycle
USB	Universal Serial Bus
XP	Extreme Programing

CHAPTER 1

INTRODUCTION

1.1 Introduction

In the information technology era, there is a huge influence of high technology and artificial intelligence when creating new software products in the whole world. This brand new method provides an effective way from bringing high quality software products to the end user. Many fields also rely on this method, especially in the Research and Development (R&D) area. As an evidence, there are many manual processes has been taking place by certain software products or artificial intelligence. Basically, every created product is operating by the combination of hardware and software to implement each feature (Perrouin G. et al. 2011). There is a closely relationship between hardware and software, both are playing an important roles to avoid failure exists.

Failure of products will happen when a human action produces some error or bug in the software and this will lead to the defects which will cause a failure occur when executed. This problem will cause serious damage of system function, and will involve higher cost and loss of time especially for a critical system. Therefore, software testing takes first priority in any Software Development Life Cycle (SDLC) to make sure the quality of software and to prevent the failure of the software.

Software testing defined as the process of executing a program on finding possible errors and validating the software or system against its specification (Myers, G. J., Badgett, T., & Sandler C. 1979). From the studies of seven principles, we know that exhaustive testing is impossible to execute all the test cases for a real software product (Wang, S., Ali, S., & Gotlieb, A. 2013). A complete testing or test for everything is impossible because there are many possible combinations of inputs and pre-condition test case for software.

Pairwise testing is an effective combinatorial method used to minimize the number of the test case that needs to inputs to a system which interactions between two input parameters values (McCaffrey, J. D. 2010). This strategy will be generating test cases that cover all the possible combinations to include the test data and to reduce the possibilities of faults due to interaction (Perrouin, G. et al. 2011). There are many pairwise testing strategies are available in the industry such as Ant Colony Algorithm (ACA) (Shiba, T., Tsuchiya, T. & Kikuno, T. 2004), Automatic Efficient Test Generator (AETG) (Cohen, D. M., Dalal, S. R., Fredman, M. L. & Patton, G. C. 1997), Genetic Algorithm (GA), Harmony Search (HS) Algorithm, In-Parameter-Order (IPO) (Lei, Y. & Tai, K. C. 1998), Intersection Residual Pair Set (IRPS) (Younis, M. I., Zamli, K. Z. & Isa, N. A. M. 2008), Simulated Annealing (SA) algorithm, All-Pairs, and so on. In Chapter 2, several existing strategies such as AETG, GA, IPO and IRPS will be elaborated in details.

1.2 Problem Statement

The main outcome for the software testing is finding defects of the existing software product. Table 1 shows the features on a laptop which consist of different Operating System (OS), Processors, System Type, Random Access Memory (RAM), Graphics Card, Battery, Universal Serial Bus (USB), Hard Drives, Screen Resolutions and Keyboard. Based on Table 1, there are consists of 10 parameters, which each parameter have two values. Therefore, exhaustive testing are happen due to 2 x 2 x 2 x 2 x 2 x 2 x 2 x 2 x 2 x 2 = 2^{10} = 1024 possible combinations need to be test to cover all the test cases. If one test case needs 5 minutes to be tested, the total minutes to complete the test will be 5120 minutes or approximately using 3 days to complete all test cases.

Table 1 Parameters and Values of Laptop Features

Parameters	Values	
OS	Windows	Linux
PROCESSORS	Intel	AMD
SYSTEM TYPE	64 bits	32 bits
RAM	2 GB	4 GB
GRAPHICS CARD	Yes	No
BATTERY	Built In	External
USB	2.0	3.0
HARD DRIVES	500 GB	1 TB
SCREEN RESOLUTION	1024×768	1280×800
KEYBOARD	With Number Pad	Without Number Pad

1.3 Goal and Objectives

The main goal of this research is to develop a Pairwise Test Data Generation Tool based on Simulated Annealing algorithm which given a name as PSAT. Following are the several objectives to achieve the goal:

i) To study the pairwise testing generation by reviewing the existing pairwise testing strategies.
ii) To apply Simulated Annealing (SA) algorithm into PSAT.
iii) To evaluate and compare the performance of proposed PSAT against with other existing strategies in term of test size.

1.4 Scope

The PSAT will be developing by using the NetBeans8.0.2 with JFrame and Java Development Kit (JDK) 8.0. Following are the scope of the research:

i) A pairwise tool with design of Graphics User Interface (GUI).
ii) A pairwise tool that adopting SA algorithm.
iii) A pairwise tool that consisting specifies values for each parameter.

1.5 Thesis Organization

This research consists of five (5) chapters that discuss the main detail in each chapter. Chapter 1 is Introduction. This chapter discusses the research introduction that will be done with including the problem statement, goal, objectives, and scope for PSAT.

Chapter 2 is Literature Review. In this chapter, we will discuss the existing research and literature review that related to the research.

Chapter 3 is Methodology. The overall approaches and framework of the research will be discussed detail in this chapter. The method, techniques or approaches will be shown on this chapter also.

Chapter 4 is Design, Implementation and Result Discussion of PSAT. In this chapter, a PSAT design will be illustrated based on the selection algorithm and implemented into PSAT. The detail of the implementation process that involved will be discussed. Lastly, the final test cases will be analyzed and evaluated whether the development is success or failure. The result also used to compare with other strategies.

Chapter 5 is Conclusion. All research that has done will be summarizing in this chapter. Besides that, the future work and alternative way to improve the research also state in this chapter.

1.6 Summary

This chapter discussed the introduction of the research of PSAT: Pairwise Test Data Generation Tool by adopting the Simulated Annealing (SA) algorithm. Problem statement, goals and objectives, scope and thesis organization are the content of this chapter. Next chapter will discuss the existing pairwise testing strategies that should be review to development our PSAT.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

In the Chapter 1, we discuss about the research introduction which included the problem statement, goals and objectives, and scope. In this Chapter, survey of the relevant literature review will be carried out to gain idea to design the PSAT. In particular, the pairwise testing techniques and existing pairwise testing strategy will be elaborated to justify the current work.

2.2 Overview

There are many existing strategy has been published and released to industry to be used in different field. In order to achieve the goal of this research, pairwise testing will be discussed in details.

2.2.1 Pairwise Testing

Pairwise testing is an effective combinatorial method used to minimize the number of the test case that needs to inputs to a system which interactions between two input parameters values (McCaffrey, J. D. 2009). In pairwise testing, a test suite will be generated to covers all the possible combination that has consists of the test data values for each pair of parameters. To understand the pairwise testing, an example will be shown with a system with four parameters which is Router, Browser, Web Server and Database Server. Each parameter of the system will consist of different values such as Router has Cisco and Huawei, Browser has Internet Explorer and Google Chrome, Database Server has Oracle and SQL Server and Web Server has Apache and Jboss as values as shown in Figure 1.

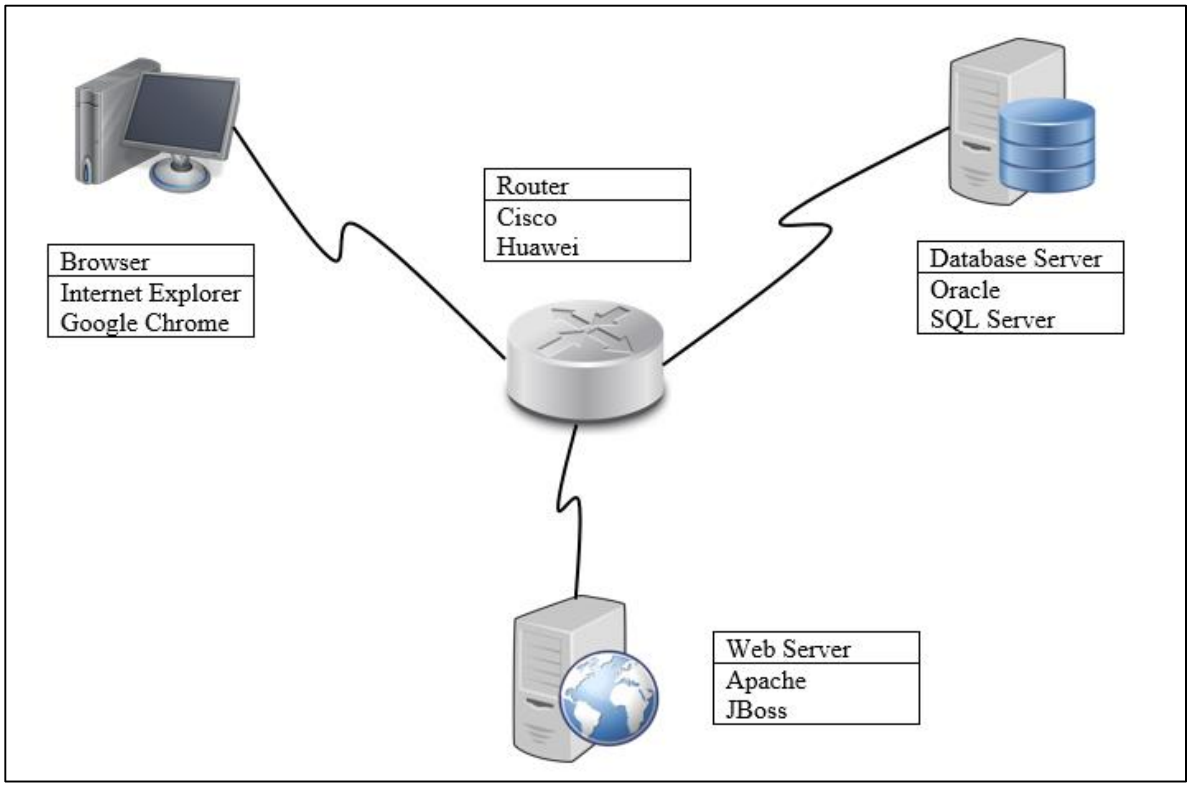


Figure 1 A System Consists of Four Parameters

Based on Figure 1, we assign the input variables into unknown term which are consists of combination of alphabet and numeric number. The result is shown on Table 2.

Table 2 Input Variables and Values of Four Parameters

Base Values	**Input Variables**			
Parameter	Router (P1)	Browser (P2)	Database Server (P3)	Web Server (P4)
Parameter Values	Cisco (A1)	Internet Explorer (B1)	Oracle (C1)	Apache (D1)
	Huawei (A2)	Google Chrome (B2)	SQL Server (C2)	Jboss (D2)

From the Table 2, the four parameters which are Router, Browser, Database Server and Web Server are assigning as Parameter 1 (P1), Parameter 2 (P2), Parameter 3 (P3) and Parameter 4 (P4). The values of each parameter also are assigning with the term of A1, A2, B1, B2, C1, C2, D1 and D2 which is combination of alphabet and numeric number. The four inputs variables are consist of two selections from the system respectively. In this situation, there are $2^4 = 16$ exhaustive combinations to cover all the possible test data and the result is shown in Table 3.

Table 3 Exhaustive Combinations of P1, P2, P3 and P4

Base Values	**Input Variables**			
Parameters	P1	P2	P3	P4
Parameter Values	A1	B1	C1	D1
	A2	B2	C2	D2
Exhaustive Combinations				
T1	A1	B1	C1	D1
T2	A1	B1	C1	D2
T3	A1	B1	C2	D1
T4	A1	B1	C2	D2
T5	A1	B2	C1	D1
T6	A1	B2	C1	D2
T7	A1	B2	C2	D1
T8	A1	B2	C2	D2
T9	A2	B1	C1	D1
T10	A2	B1	C1	D2
T11	A2	B1	C2	D1
T12	A2	B1	C2	D2
T13	A2	B2	C1	D1
T14	A2	B2	C1	D2
T15	A2	B2	C2	D1
T16	A2	B2	C2	D2

Pairwise testing techniques have been used to reduce the exhaustive combinations for this system. By using this technique, a 2-way possible combinations produce P1P2, P1P3, P1P4, P2P3, P2P4 and P3P4 as the combinations. For P1P2 combination, the test case has been reduced to four test cases which only consider P1 and P2 as a pair. P3 and P4 are randomly assigned a value as shown in Table 4.

Table 4 2-way Combination for P1P2

Base Values	**Input Variables**			
	P1	P2	P3	P4
	A1	B1	C1	D1
	A2	B2	C2	D2
2-way Combinations for P1P2				
Test Case 1	A1	B1	C1	D1
Test Case 2	A1	B2	C2	D2
Test Case 3	A2	B1	C1	D1
Test Case 4	A2	B2	C2	D2

The second combination is P1P3. For this combination, the test case also been reduced become four test cases and only consider P1 and P3 as a pair. P2 and P4 are randomly assigned a value as shown in Table 5.

Table 5 2-way Combination for P1P3

Base Values	**Input Variables**			
	P1	P2	P3	P4
	A1	B1	C1	D1
	A2	B2	C2	D2
2-way Combinations for P1P3				
Test Case 1	A1	B1	C1	D1
Test Case 2	A1	B2	C2	D2
Test Case 3	A2	B1	C1	D1
Test Case 4	A2	B2	C2	D2

The third combination is P1P4. For this combination, the total test size also reduced become four test cases and only considers P1 and P4 as a pair. P2 and P3 are randomly assigned a value as shown in Table 6.

Table 6 2-way Combination for P1P4

Base Values	**Input Variables**			
	P1	P2	P3	P4
	A1	B1	C1	D1
	A2	B2	C2	D2
2-way Combination for P1P4				
Test Case 1	A1	B1	C1	D1
Test Case 2	A1	B2	C2	D2
Test Case 3	A2	B1	C1	D1
Test Case 4	A2	B2	C2	D2

The fourth combination is P2P3. For this combination, the total test size also reduced become four test cases and only considers P2 and P3. The value is randomly assigned for P1 and P4 as shown in Table 7.

Table 7 2-way Combination for P2P3

Base Values	**Input Variables**			
	P1	P2	P3	P4
	A1	B1	C1	D1
	A2	B2	C2	D2
2-way Combination for P2P3				
Test Case 1	A1	B1	C1	D1
Test Case 2	A1	B1	C2	D2
Test Case 3	A2	B2	C1	D1
Test Case 4	A2	B2	C2	D2

The fifth combination is P2P4. For this combination, the total test size also reduced become four test cases and only considers P2 and P4. The value is randomly assigned for P1 and P3 as shown in Table 8.

Table 8 2-way Combination for P2P4

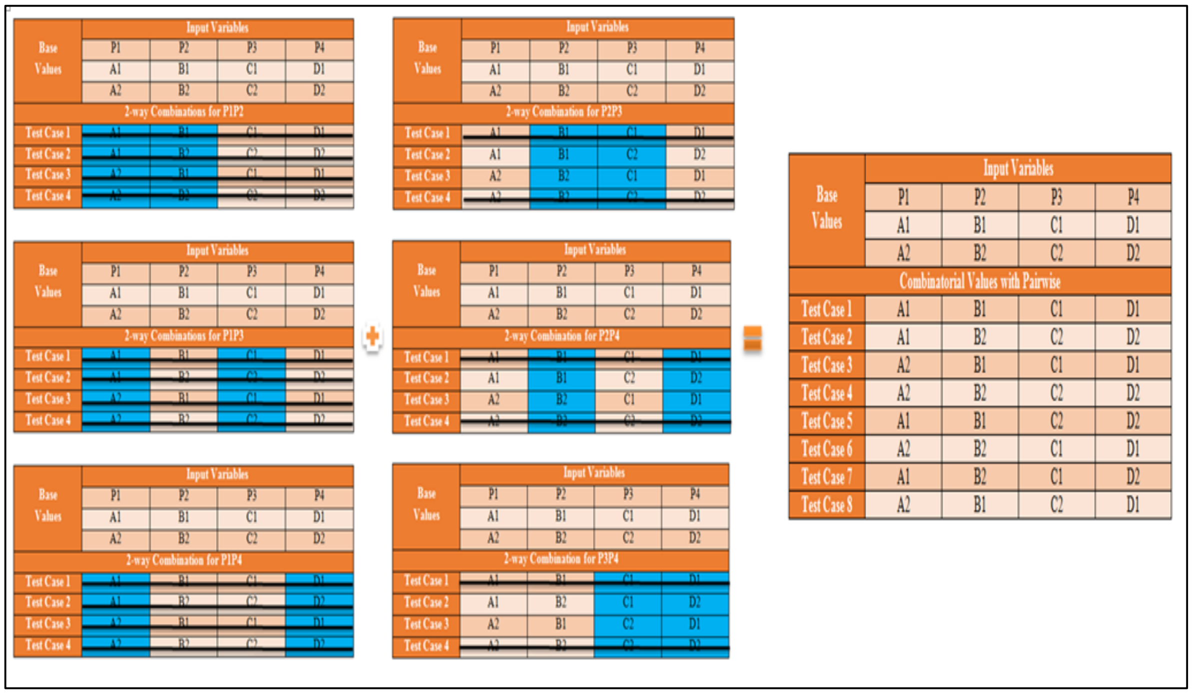
Base Values	**Input Variables**			
	P1	P2	P3	P4
	A1	B1	C1	D1
	A2	B2	C2	D2
2-way Combination for P2P4				
Test Case 1	A1	B1	C1	D1
Test Case 2	A1	B1	C2	D2
Test Case 3	A2	B2	C1	D1
Test Case 4	A2	B2	C2	D2

Lastly, the combination is P3P4. For this combination, the total test size also reduced become four test cases and only considers P3 and P4. The value is randomly assigned for P1 and P2 as shown in Table 9.

Table 9 2-way Combination for P3P4

Base Values	**Input Variables**			
	P1	P2	P3	P4
	A1	B1	C1	D1
	A2	B2	C2	D2
2-way Combination for P3P4				
Test Case 1	A1	B1	C1	D1
Test Case 2	A1	B2	C1	D2
Test Case 3	A2	B1	C2	D1
Test Case 4	A2	B2	C2	D2

After merging the combinations of P1P2, P1P3, P1P4, P2P3, P2P4 and P3P4, the similar interaction of the combinations are created as a new test case and save into the final test suite. After save into the final test suite, the interaction are deleted and do not count for the next test case until all the interaction is covered. The result of this system shows that the final test suite has been reducing from 16 test cases to 8 cases which reduction of 50%. Figure 2 shown the total test cases after merge all the combinations.



Base Values	Input Variables			
	P1	P2	P3	P4
	A1	B1	C1	D1
	A2	B2	C2	D2
2-way Combinations for P1P2				
Test Case 1	~~A1~~	~~B1~~	~~C1~~	~~D1~~
Test Case 2	~~A1~~	~~B2~~	~~C2~~	~~D2~~
Test Case 3	~~A2~~	~~B1~~	~~C1~~	~~D1~~
Test Case 4	~~A2~~	~~B2~~	~~C2~~	~~D2~~

Base Values	Input Variables			
	P1	P2	P3	P4
	A1	B1	C1	D1
	A2	B2	C2	D2
2-way Combination for P2P3				
Test Case 1	~~A1~~	~~B1~~	~~C1~~	~~D1~~
Test Case 2	A1	B1	C2	D2
Test Case 3	A2	B2	C1	D1
Test Case 4	~~A2~~	~~B2~~	~~C2~~	~~D2~~

Base Values	Input Variables			
	P1	P2	P3	P4
	A1	B1	C1	D1
	A2	B2	C2	D2
2-way Combinations for P1P3				
Test Case 1	~~A1~~	~~B1~~	~~C1~~	~~D1~~
Test Case 2	~~A1~~	~~B2~~	~~C2~~	~~D2~~
Test Case 3	~~A2~~	~~B1~~	~~C1~~	~~D1~~
Test Case 4	~~A2~~	~~B2~~	~~C2~~	~~D2~~

Base Values	Input Variables			
	P1	P2	P3	P4
	A1	B1	C1	D1
	A2	B2	C2	D2
2-way Combination for P2P4				
Test Case 1	~~A1~~	~~B1~~	~~C1~~	~~D1~~
Test Case 2	A1	B1	C2	D2
Test Case 3	A2	B2	C1	D1
Test Case 4	~~A2~~	~~B2~~	~~C2~~	~~D2~~

Base Values	Input Variables			
	P1	P2	P3	P4
	A1	B1	C1	D1
	A2	B2	C2	D2
2-way Combination for P1P4				
Test Case 1	~~A1~~	~~B1~~	~~C1~~	~~D1~~
Test Case 2	~~A1~~	~~B2~~	~~C2~~	~~D2~~
Test Case 3	~~A2~~	~~B1~~	~~C1~~	~~D1~~
Test Case 4	~~A2~~	~~B2~~	~~C2~~	~~D2~~

Base Values	Input Variables			
	P1	P2	P3	P4
	A1	B1	C1	D1
	A2	B2	C2	D2
2-way Combination for P3P4				
Test Case 1	~~A1~~	~~B1~~	~~C1~~	~~D1~~
Test Case 2	A1	B2	C1	D2
Test Case 3	A2	B1	C2	D1
Test Case 4	~~A2~~	~~B2~~	~~C2~~	~~D2~~

Base Values	Input Variables			
	P1	P2	P3	P4
	A1	B1	C1	D1
	A2	B2	C2	D2
Combinatorial Values with Pairwise				
Test Case 1	A1	B1	C1	D1
Test Case 2	A1	B2	C2	D2
Test Case 3	A2	B1	C1	D1
Test Case 4	A2	B2	C2	D2
Test Case 5	A1	B1	C2	D2
Test Case 6	A2	B2	C1	D1
Test Case 7	A1	B2	C1	D2
Test Case 8	A2	B1	C2	D1

Figure 2 Merging of the P1, P2, P3 and P4

2.3 Survey of Existing Pairwise Strategies

In this topic, the surveys will be done based on the existing work on pairwise testing. This existing pairwise testing is difference in two approaches which are one test at a time (OTAT) and one parameter at a time (OPAT). We will review some existing techniques for each approach. In one test at a time approaches, we will review the GA (Mitchell, M. 1995) and AETG (Cohen, D. M., Dalal, S. R., Fredman, M. L. & Patton, G. C. 1997). While in one parameter at a time approaches, we will review on IPO (Lei, Y. & Tai, K. C. 1998) and IRPS (Younis, M. I., Zamli, K. Z. & Isa, N. A. M. 2008). Figure 3 show the design of the existing pairwise strategies.

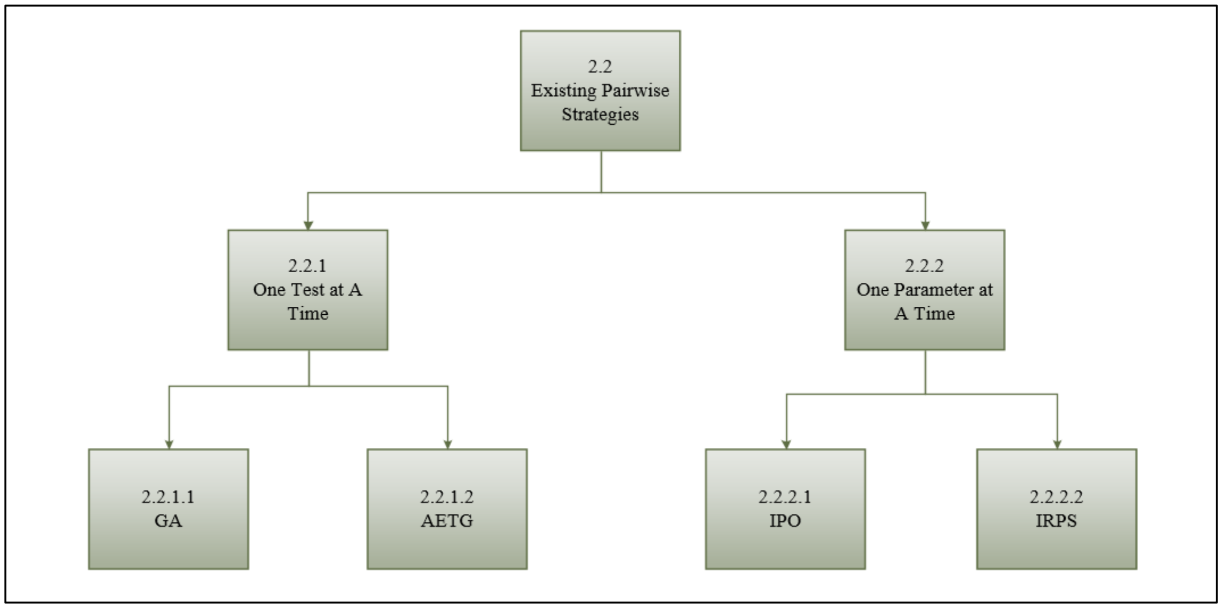


Figure 3 Existing Pairwise Strategies

2.3.1 One Test at a Time (OTAT)

OTAT is an approach that using by several existing pairwise testing strategies. OTAT is meant to generate a test case on one time in each iteration. The rule of OTAT is generating all the test case one by one until all test cases have been covering all the possible interaction. There are many existing strategies in this approaches such as AETG (Cohen, D. M., Dalal, S. R., Fredman, M. L. & Patton, G. C. 1997), AETG2, SA, GA, ACA (Shiba, T., Tsuchiya, T. & Kikuno, T. 2004), All-Pairs, G2Way, Jenny (Younis, M. I., Zamli, K. Z. & Isa, N. A. M. 2008), SA_SAT, mAETG_SAT (Alsewari, A. R. A. & Zamli, K. Z. 2012), Pairwise Independent Combinatorial Testing (PICT) (Ahmed. B. S. & Zamli, K. Z. 2011), TestCover, LAHC, HSS and BA_PTCS (Alsariera, Y. A., Alsewari, A. R. A. & Zamli, K. Z. 2015). Here, we discussed by selecting one of the nature based algorithms such as GA (Mitchell, M. 1995) and one of the first strategies based on pure computational techniques which are AETG.

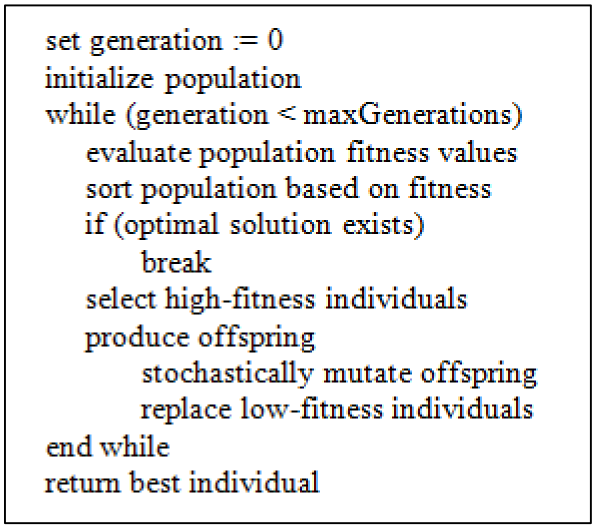
2.3.1.1 Genetic Algorithm (GA)

GA is a search heuristic that mimic the processes of biological evolution (Mitchell, M. 1995). GA is used on finding best solutions to optimization-related problem and search problem (Flores, P. & Cheon, Y. 2011). In GA, there are three main elements that consider inside the GA which are chromosome encoding, fitness function and genetic operations. In chromosome encoding, an individual will be encoded as a chromosome that have one value of each parameter (Shiba, T., Tsuchiya, T. & Kikuno, T. 2004).

Mostly, each chromosome of the current population will assign a score (fitness) from the fitness function. The fitness of each chromosome will measure the standard of the chromosome to solve the problem. The fitness function is used to measure the potential of individual to find out the good one. The individual that have higher fitness will have more opportunity to stay in a population.

In the genetic operations, there are three types of operators which are selection, crossover, and mutation (Mitchell, M. 1995):

a)	Selection	: Used to selects the chromosomes for reproduction.
b)	Crossover	: Combinations of two chromosomes to create two offspring.
c)	Mutation	: Randomly changes or modified the chromosomes.



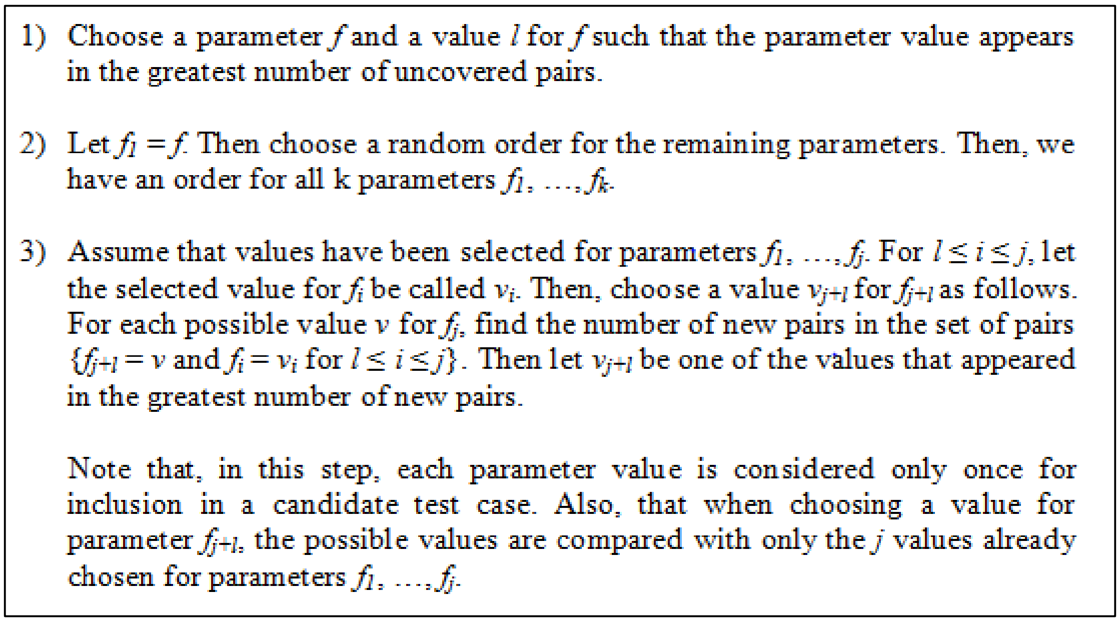
```
set generation := 0
initialize population
while (generation < maxGenerations)
    evaluate population fitness values
    sort population based on fitness
    if (optimal solution exists)
        break
    select high-fitness individuals
    produce offspring
        stochastically mutate offspring
        replace low-fitness individuals
end while
return best individual
```

Figure 4 The Genetic Algorithm

Normally, produce a new generation of the population as same as shown in Figure 4 (McCaffrey, J. D. 2009). Firstly, select a pair of parent chromosomes from the population based on the fitness functions. Secondly, two offspring will be produced from the pair of parent chromosomes through the crossover. Third, randomly change the fitness functions of the two offspring by using mutation operator and replace the result in the new population. This step is repeated to generate the new population until the stopping conditions are met.

2.3.1.2 Automatic Efficient Test Generator (AETG)

AETG is a strategy that used to generate efficient test cases based on the combinatorial design (Cohen, D. M., Dalal, S. R., Fredman, M. L. & Patton, G. C. 1997). AETG is used a greedy algorithm to generate a test set by adding one test case which will cover all the n-way combinations of parameter (Shiba, T., Tsuchiya, T. & Kikuno, T. 2004). Since the number of test set required to cover all the n-way combinations, therefore the size of test set are grows logarithmically in the number of test parameters to generate an efficient test case. Figure 5 shows the algorithm of AETG (Cohen et al. 1997).



1) Choose a parameter f and a value l for f such that the parameter value appears in the greatest number of uncovered pairs.

2) Let $f_1 = f$. Then choose a random order for the remaining parameters. Then, we have an order for all k parameters $f_1, \ldots, f_k$.

3) Assume that values have been selected for parameters $f_1, \ldots, f_j$. For $1 \le i \le j$, let the selected value for f_i be called v_i. Then, choose a value v_{j+1} for f_{j+1} as follows. For each possible value v for f_j, find the number of new pairs in the set of pairs $\{f_{j+1} = v$ and $f_i = v_i$ for $1 \le i \le j\}$. Then let v_{j+1} be one of the values that appeared in the greatest number of new pairs.

   Note that, in this step, each parameter value is considered only once for inclusion in a candidate test case. Also, that when choosing a value for parameter f_{j+1}, the possible values are compared with only the j values already chosen for parameters $f_1, \ldots, f_j$.

Figure 5 The AETG Greedy Algorithm

In AETG algorithm, is same as an incremental method which starts generate an empty test set by adding one test case until meet 100% coverage for the interaction. This AETG algorithm help to find the optimal test case because it assumes to find test case that covers the maximum number of combinations which consists of many possible test cases (Cohen, D. M., Dalal, S. R., Fredman, M. L. & Patton, G. C. 1997).

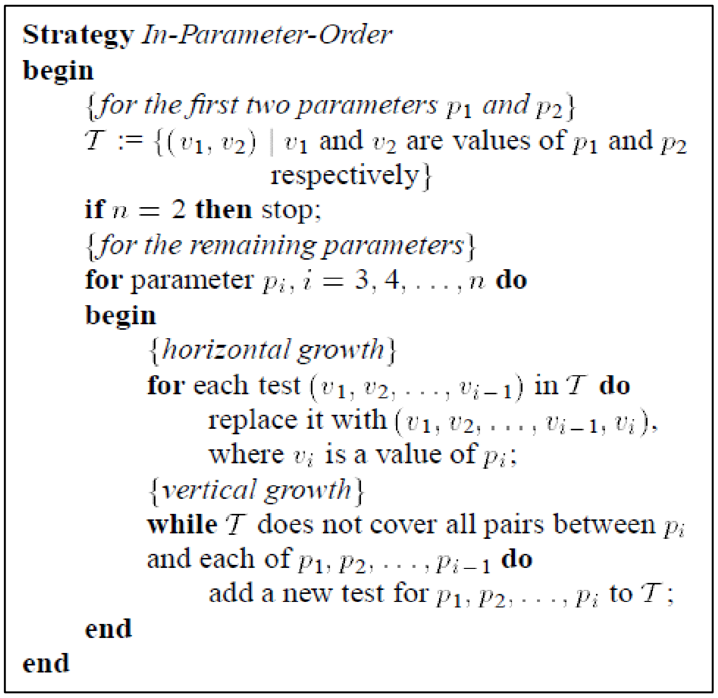
2.3.2 One Parameter at a Time (OPAT)

OPAT is another approach that uses to generate all test cases by starting from the first pairs then add one parameter's values in each iteration until all pairs are covered. In this way, we have selected one of the pure computational techniques which are IPO (Lei, Y. & Tai, K. C. 1998) and one of the improvements of computational techniques like IRPS (Younis, M. I., Zamli, K. Z. & Isa, N. A. M. 2008).

2.3.2.1 In-Parameter-Order (IPO)

IPO is a test generation strategy for pairwise testing. This strategy is used to generate the pairwise test set for the values of the first two parameters follow by extending the test suite for the first three parameters (Lei, Y., Kacker, R., Kuhn, D. R., Okun, V., & Lawrence, J. 2008). The step will be continues for every additional another parameter until all parameters are included in the test suite (Lei, Y. & Tai, K. C. 1998).

There is two algorithms used by IPO to extend test suite for each additional of new parameters. The first algorithm that been used is horizontal growth that able to add the value of the new parameters by extending the existing test suite. While the second algorithm which is vertical growth that used to satisfy between the test suite and pairwise coverage for the value of the new parameters by adding new test cases. Figure 6 shows the IPO algorithm (Lei, Y. & Tai, K. C. 1998).



```
Strategy In-Parameter-Order
begin
        {for the first two parameters $p_1$ and $p_2$}
        $T := \{(v_1, v_2) \mid v_1$ and $v_2$ are values of $p_1$ and $p_2$
                        respectively$\}$
        if $n = 2$ then stop;
        {for the remaining parameters}
        for parameter $p_i, i = 3, 4, \ldots, n$ do
        begin
                {horizontal growth}
                for each test $(v_1, v_2, \ldots, v_{i-1})$ in $T$ do
                        replace it with $(v_1, v_2, \ldots, v_{i-1}, v_i)$,
                        where $v_i$ is a value of $p_i$;
                {vertical growth}
                while $T$ does not cover all pairs between $p_i$
                and each of $p_1, p_2, \ldots, p_{i-1}$ do
                        add a new test for $p_1, p_2, \ldots, p_i$ to $T$;
        end
end
```

Figure 6 The IPO Algorithm

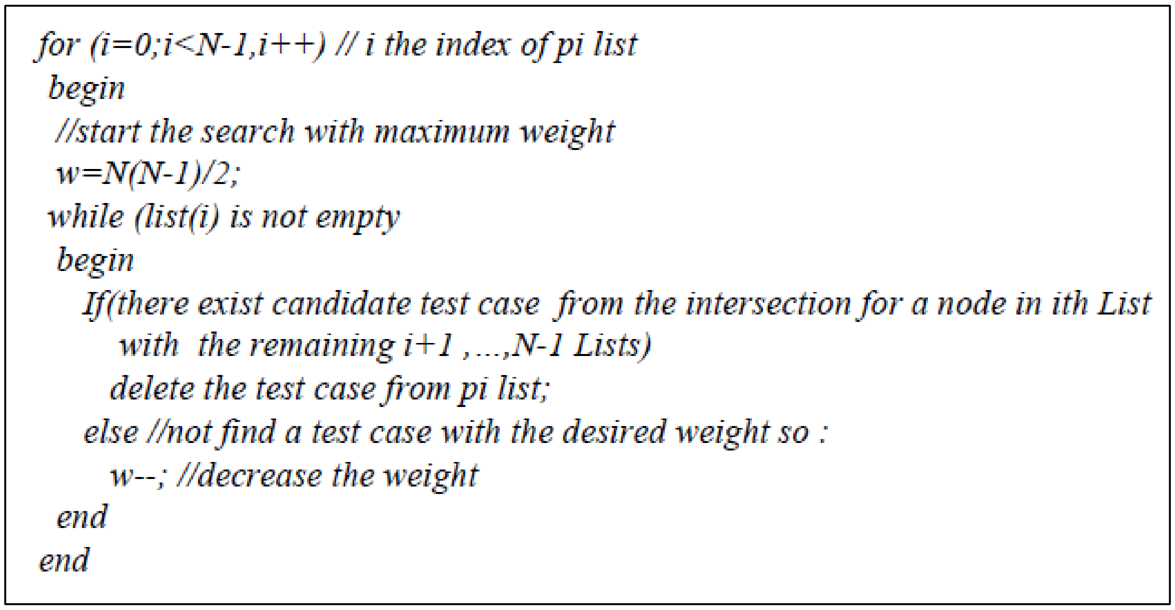
Lately, the improvement of IPO strategy to become In-Parameter-Order-General (IPOG), In-Parameter-Order-General with D- construction (IPOG-D) (Lei, Y., Kacker, R., Kuhn, D. R., Okun, V. & Lawrence, J. 2007), IPOG-F which is In-Parameter-Order-General with FireEye and IPOG-F2 (Forbes, M., Lawrence, J., Lei, Y., Kacker, R. N. & Kuhn, D. R. 2008) that used to optimize the test cases and reduce the execution time (Alsewari, A. R. A. & Zamli, K. Z. 2012).

2.3.2.2 Intersection Residual Pair Set (IRPS)

IRPS is another efficient pairwise strategies used to test data generation. Development of IRPS is a way used to generate the pairwise test set with optimum size (Younis, M. I., Zamli, K. Z. & Isa, N. A. M. 2008). There are several steps to conduct when generating the test set using IRPS:

1) Step 1 : Save all pairs into a compact linked list called Pi after generating the test pairs.
2) Step 2 : Search in the Pi list and find the desired weight from the candidate case, this candidate case with become a test case and will deleted form the Pi list.
3) Step 3 : Continue with Step 2 until there are nothing inside the Pi list.

IRPS will generate and stored all pairs into the linked list, Pi by using array. Each linked list consists of the nodes that equal to the number of the parameters. Weight of the candidate test case is important to IRPS because IRPS need to calculate the combinations which cover all pairs of the candidate test case. The candidate will treat a test case based on the desired weight criteria, or else the intersection between the node and other nodes will continue to generate. The repeated variables in each node will be deleted by using delete operation in IRPS. Following Figure 7 is the search algorithm used by the IRPS (Younis, M. I. et al. 2008).



```
for (i=0;i<N-1,i++) // i the index of pi list
 begin
  //start the search with maximum weight
  w=N(N-1)/2;
 while (list(i) is not empty
  begin
    If(there exist candidate test case  from the intersection for a node in ith List
       with  the remaining i+1 ,...,N-1 Lists)
      delete the test case from pi list;
    else //not find a test case with the desired weight so :
      w--; //decrease the weight
  end
 end
```

Figure 7 The Search Algorithm for IRPS

2.4 Analysis of Existing Pairwise Strategies

After studies of all existing pairwise strategies as mentioned above, we know that the existing strategies separate in two different approaches which are OTAT and OPAT. There are different comparisons between these two approaches. For OTAT, a complete test case will be generated by cover all the interaction elements while the OPAT is tested using selected element parameters of the test cases by adding one parameter of each time to produce sufficient test case (Alsewari, A. R. A. & Zamli, K. Z. 2012). For each existing pairwise strategies, the n-way are t = 2.

Table 10 Summary Analysis of the Existing Pairwise Strategies

Approaches	Existing Strategies	n-way	Deterministic	GUI
One Test at a Time (OTAT)	GA	t = 2	×	×
	AETG	t = 2	×	✓
One Parameter at a Time (OPAT)	IPO	t = 2	✓	×
	IRPS	t = 2	✓	×

From the Table 10 above, we know that GA and AETG is a non-deterministic approach. Non-deterministic means that the generated test cases cannot be determined or another define is the same input parameter may appear in the different test cases (Grindal, M., Offutt, J. & Andler, F. 2004). For IPO and IRPS, they are classified to deterministic approaches which are always generate same test case result by using a specific input parameter (Grindal, M., Offutt, J. & Andler, F. 2004). Furthermore, some of the existing pairwise strategies do not design with Graphics User Interface such as GA, IPO and IRPS. Therefore, to adopt the design of PSAT, we proposed a Pairwise Test Data Generation Tool by using Simulated Annealing Algorithm. In the Chapter 4, this generator tool will be explained in more details.

2.5 Summary

This chapter has discussed the pairwise testing and previous existing pairwise testing strategies. The existing pairwise testing strategy that cover in this chapter is Genetic Algorithm (GA), Automatic Efficient Test Generator (AETG), In-Parameter-Order (IPO), and Intersection Residual Pair Set (IRPS). In general, all pairwise strategies have the same objective which is to minimize the test case, identify the defects and increase the coverage of interaction.

CHAPTER 3

METHODOLOGY

In Chapter 2, we have been discussed in details about the several existing pairwise testing strategies. Therefore, in this chapter we will discuss about the methodology that we will use to finish up this research. Rapid Application Development (RAD) whereas one of the existing methodology will be used as a guideline to build the Pairwise Test Data Generation Tool based on the Simulated Annealing algorithm (PSAT). There are several stage consists in RAD and the importance for each stages will be explained in details in this chapter.

3.1 Introduction

The proposed PSAT is designed by implementing the SA algorithm. As we discuss in the previous chapter, the goals of designing these generator tools is to minimize the number of test cases for an input system. Therefore, SA algorithm is implementing in the PSAT because SA is a random search algorithm that able generated the least number of test cases for this research.

To complete this research as we planned, a methodology is needed to include into development of PSAT. The methodology is a standard framework used to plan and manage the activities inside the process development of PSAT. There is important for every project to have a methodology as a guideline to ensure the goals of the research is achieved.

There are a lot of different kind methodologies in the software development process. The most common methodologies consist of Waterfall Methodology, V-Model Methodology, Spiral Methodology, Agile Methodology, Rapid Application Development (RAD), and Extreme Programing (XP). In this research, RAD has been chosen as the methodology to develop our Pairwise Test Data Generation Tool based on Simulated Annealing Algorithm (PSAT).

3.2 Methodology

RAD is to improve the process of development which faster than the Waterfall Methodology (Martin, J. 1991). By using RAD, our research is able to produce high quality generator tools and have the potential to reduce time to complete this research. We adopt the RAD methodology and have carried out four stages which are requirement planning stage, user design stage, implementation stage and evaluation and documentation stage. Figure 8 shows the stages of the RAD model.

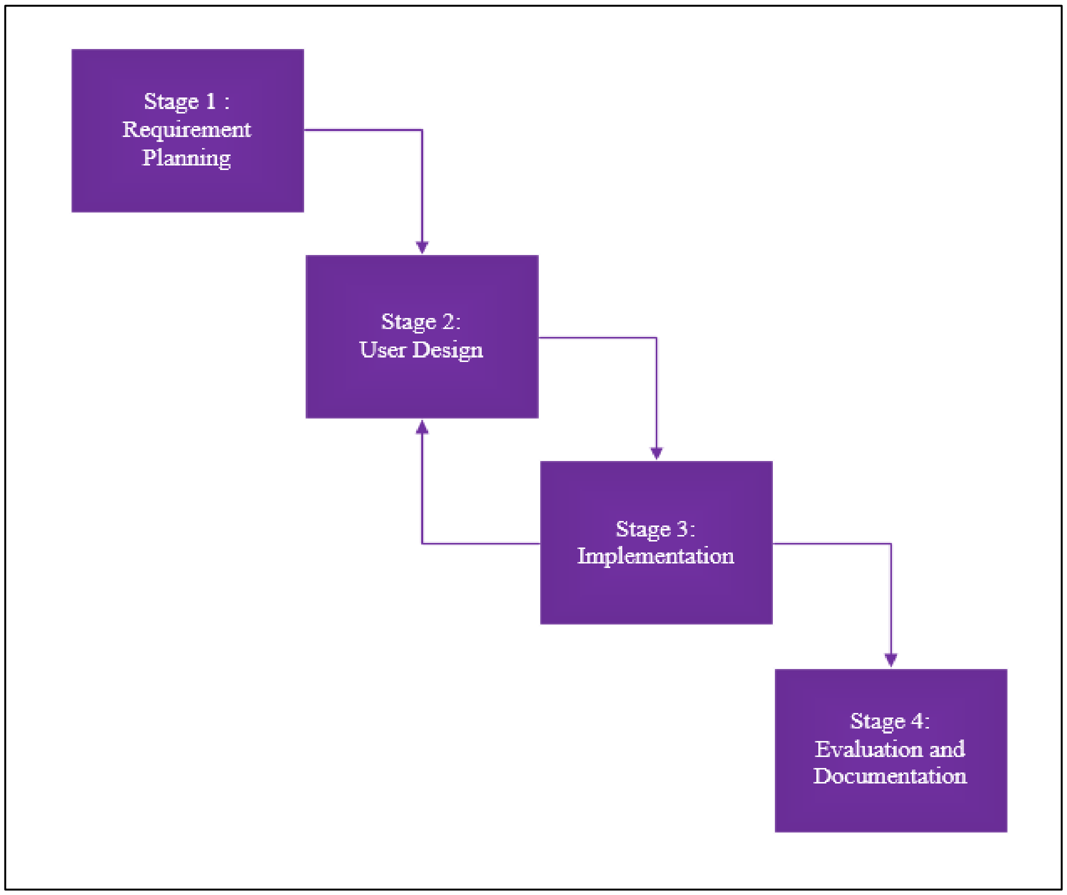

Stage 1 : Requirement Planning
Stage 2: User Design
Stage 3: Implementation
Stage 4: Evaluation and Documentation

Figure 8 Methodology Based on Rapid Application Development (RAD) Model

3.2.1 Requirement Planning

In this stage, we have analyzed and determine the existing problem that related to this research. There is an important step before we start to decide the goals and objectives of the research. The goals of this research are to develop a PSAT by adopting SA algorithm. NetBeans IDE 8.0.2 with JFrame and JDK 8.0 has been chosen as the development tool to develop these PSAT.

Besides that, the literature review has been performed to analyze and plan the idea to develop the PSAT based on SA algorithm. Research on other existing pairwise testing strategies is a must to collect more information to complete this research. The existing pairwise testing strategies including GA (Mitchell, M. 1995), AETG (Cohen, D. M., Dalal, S. R., Fredman, M. L. & Patton, G. C. 1997), IPO (Lei, Y. & Tai, K. C. 1998), and IRPS (Younis, M. I., Zamli, K. Z. & Isa, N. A. M. 2008) has been explained in details.

3.2.2 User Design

In this stage, the design of the PSAT based on JFrame will be carried out. There are four (4) techniques to design the PSAT. For the first techniques, a prototype user interface will be designed for these generation tool. The interface is used to show the required content of the generation tool to the user and request input from the user. The second technique is the interaction list generation. Here, all the possible pairs of test case will be generated according to the input from the user. Test case generation is the third technique in stage three of user design. After getting all the possible pairs, the generator tool will find out and generate the sufficient test case based on the SA algorithm. In the last techniques, the test case will be generated and map with the actual parameter value name to show to user.

3.2.3 Implementation

In this stage, the study of SA algorithm will be implemented into the PSAT because SA algorithm is the search technique for this proposed research. The SA algorithm will be created and test to make sure the algorithm is work properly and run smoothly by using the NetBeans IDE 8.0.2. The algorithm will be adopted with the user interface to generate out the sufficient test cases. The result of the generation will be collected and recorded.

3.2.4 Evaluation and Documentation

In this stages, the result of test cases that collect from stage three will be used to compared with the result that produce from other existing pairwise testing tools. The comparison of test cases is to check the performance of SA algorithm and evaluate which pairwise tool is better.

By last, the research information and result will be finalized and documented in this research. Furthermore, this documentation will become as a reference for the future use or improvement.

3.3 Hardware and Software

There are some requirements of the research to ensure the research meets the goals successfully. Therefore, a few hardware and software requirements are needed to fulfill along this research. Table 11 shows the hardware specifications and Table 12 shows the software specifications of this research.

Table 11 Hardware Specifications for Development PSAT

Hardware	Specifications	Description/Importance
Laptop	ASUS K43SD, Intel Core i5 2450M 2.5GHz, 6GB DDR3 SDRAM, 2.5" SATA 500GB 5400rpm.	Used for the whole research that include: 1. Documentation, 2. Design user interface, 3. Algorithm implementation, 4. Comparison for result.
External Hard Disk	1TB Toshiba	1. Backup project files 2. Storage
Thumb Drive	8GB Kingston DT101 G2	1. File transfer

Table 12 Software Specifications for Development PSAT

Software	Specifications	Description/Importance
Operating System	Window 10 Education 64-bit	1. Platform for research development
Development Tool	NetBeans IDE 8.0.2	1. Java language development tool
Running Environment	1. JDK 8.0, 2. JRE 8.0, 3. JFrame.	1. Implementation platform for research development 2. Design GUI
Word Processor	Microsoft Word 2013	1. Write documentation 2. Draft
Plan and Schedule	Microsoft Project 2013	1. Gantt chart planning
Diagram	Microsoft Visio 2013	1. Draw Diagram
Presentation	Microsoft Office 2013	1. Slide preparation
Document Reader	Adobe Reader	1. Read journal, article, and paper 2. Preview journal, article, and paper
Web Browser	1. Google Chrome, 2. Mozilla Firefox, 3. Internet Explorer.	1. Search information

3.4 Summary

In this chapter, we have discussed the methodology that used on our research which is Rapid Application Development (RAD). In RAD, there are separate the stage in four stages to complete our PSAT. Besides that, the hardware and software specification that include in the research also been discussed in this chapter.

CHAPTER 4

DESIGN, IMPLEMENTATION AND RESULT DISCUSSION

4.1 Introduction

In the previous chapter, we have been discussed about the Rapid Application Development (RAD) methodology that been used as a guideline to accomplish this research. In this chapter, we will comprehensively discuss the designing of PSAT and how the structure and algorithms that will be used to implement into our PSAT. After the implementation, PSAT will be used to evaluate and compare with the existing pairwise testing strategies as mentioned in Chapter 2. All the results will be recorded and documented.

4.2 Development of PSAT

In this section, a full explanation on the structure development of PSAT and how the flows on generating PSAT test cases will be discussed in details. In this research, the PSAT process has been categories into four techniques which are the user interface, interaction list generation, test case generation and mapping generation. Four of the techniques are the main important process to complete the PSAT prototype and this proposed research. Figure 9 illustrate the overview of this framework.

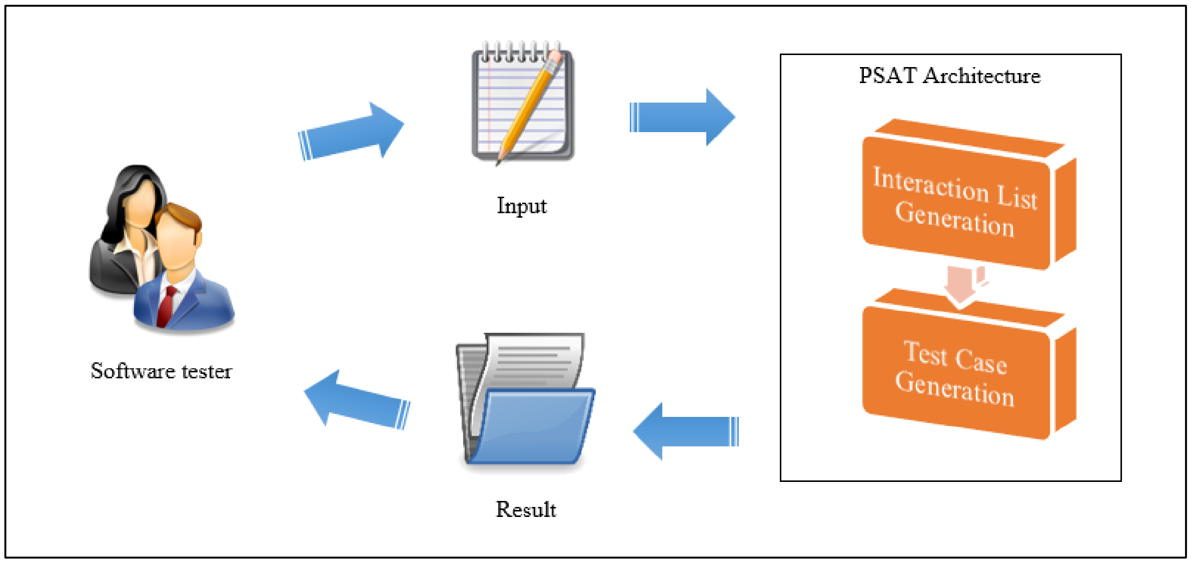


Figure 9 The Overview of PSAT Framework

Based on Figure 4.1, software tester requires entering the input parameter and the input values of each parameter through the input screen from user interface that created using JFrame. Based on the input parameter and input values, the interaction list generation will be triggered and all the possible interaction pairs will be generated. When the interaction pairs cover all the possible interaction of the input parameters, the interaction pairs will be saved into an arraylist named interaction list. As usual, each interaction pairs consists of two different values which are the combination of two different input parameters. This is because the proposed research is pairwise testing and pairwise testing is a 2-way combinatorial strategy.

In this research, SA algorithm is our main algorithm used to generate test case with randomly in test case generation. We using SA algorithm to generate the test case randomly according to the interaction list and find the weightage of each test case. When the test case covers the maximum weightage, the selected test case will be saved into the final test list. While the test case does not cover maximum weightage, the test case will be process based on probability to find out the test case and save the test case in final test list again. After test case generation find out all the sufficient test case, PSAT will return the exact result to the software tester. Figure 10 shows the flow chart of this proposed research framework.

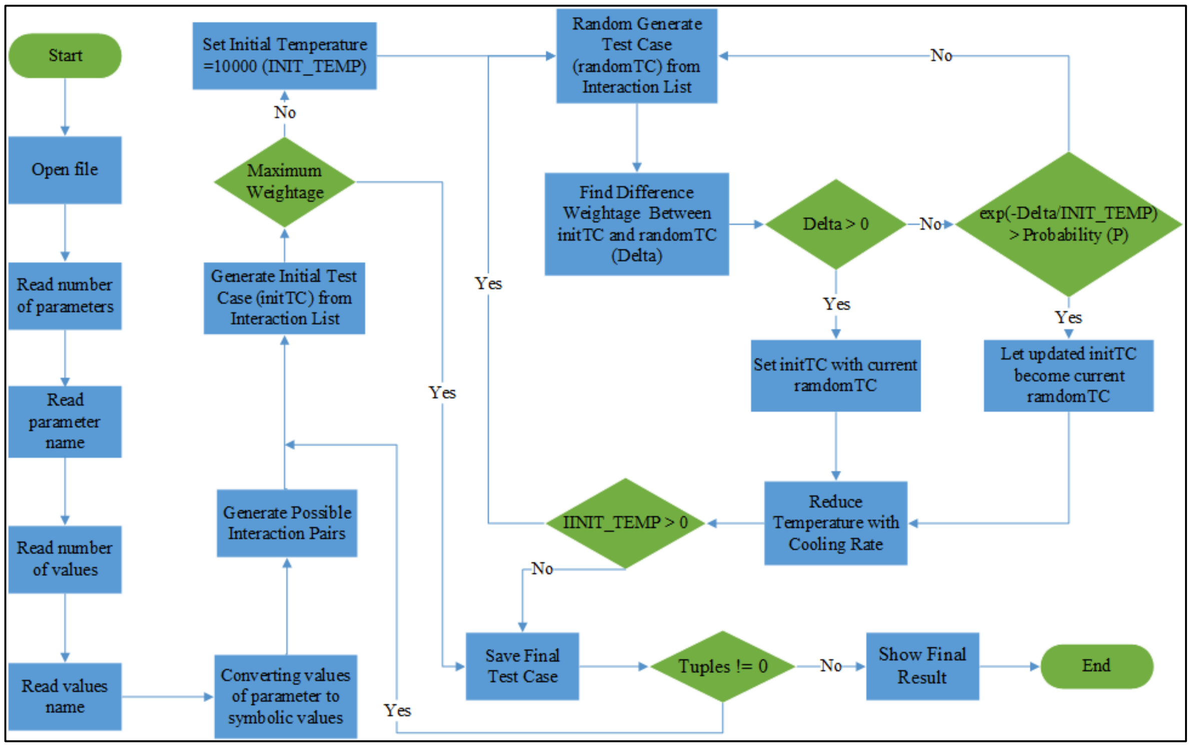
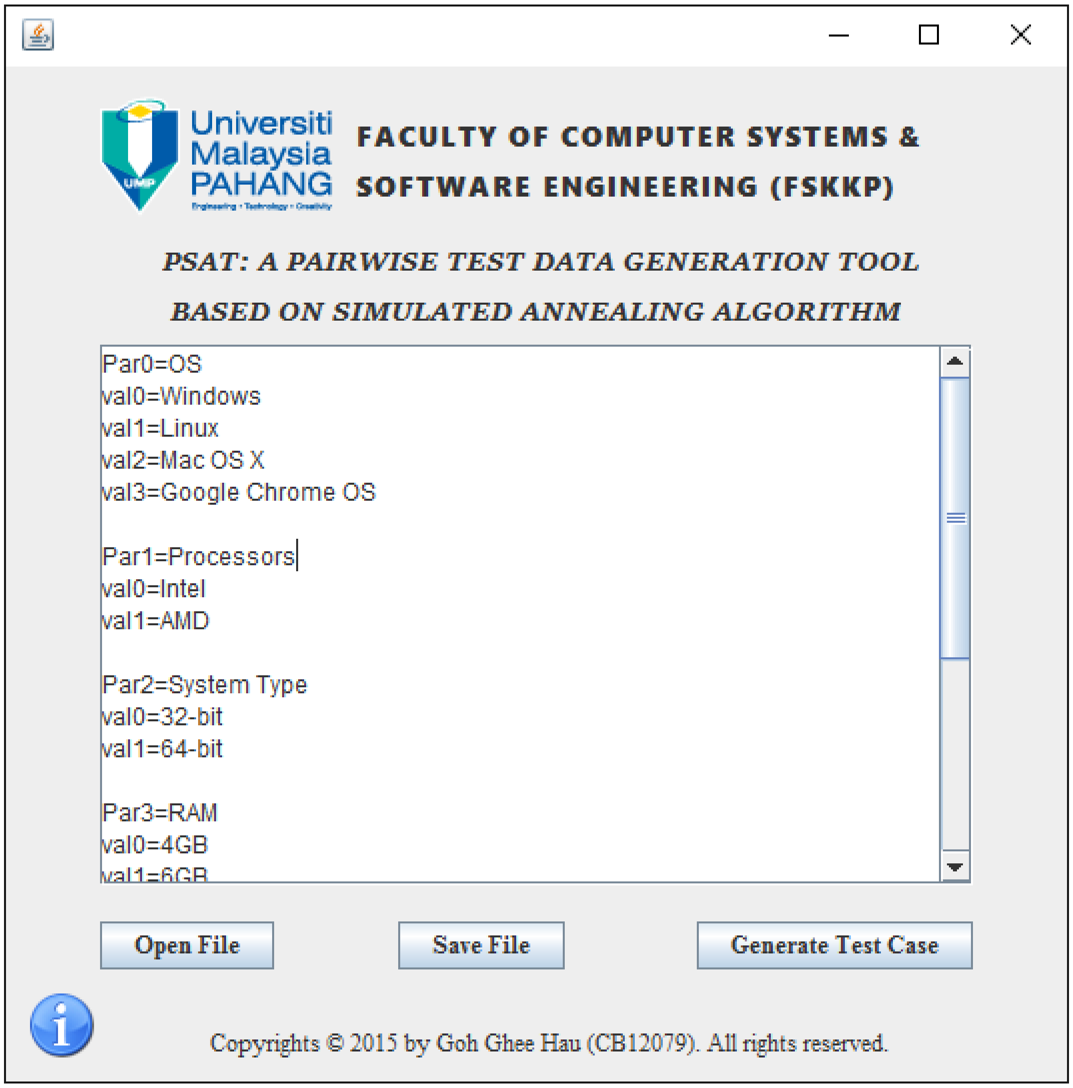

Start
Open file
Read number of parameters
Read parameter name
Read number of values
Read values name
Converting values of parameter to symbolic values
Generate Possible Interaction Pairs
Generate Initial Test Case (initTC) from Interaction List
Maximum Weightage
No
Set Initial Temperature =10000 (INIT_TEMP)
Random Generate Test Case (randomTC) from Interaction List
Find Difference Weightage Between initTC and randomTC (Delta)
Delta > 0
No
exp(-Delta/INIT_TEMP) > Probability (P)
No
Yes
Yes
Set initTC with current ramdomTC
Let updated initTC become current ramdomTC
Reduce Temperature with Cooling Rate
IINIT_TEMP > 0
Yes
No
Yes
Save Final Test Case
Tuples != 0
Yes
No
Show Final Result
End

Figure 10 Flow Chart for the Whole PSAT Framework

4.2.1 User Interface of PSAT

As we mentioned above, there are four important techniques will be used to complete the PSAT prototype. In this section, the first technique to develop this proposed research is user interface. The prototype user interface is designed using NetBeans 8.0.2 with JFrame which is an interface frame with a title and a border in NetBeans IDE. The important of the prototype user interface is as an interaction to gather input from user, show the required contents to the user and provide the expected result that user needs.

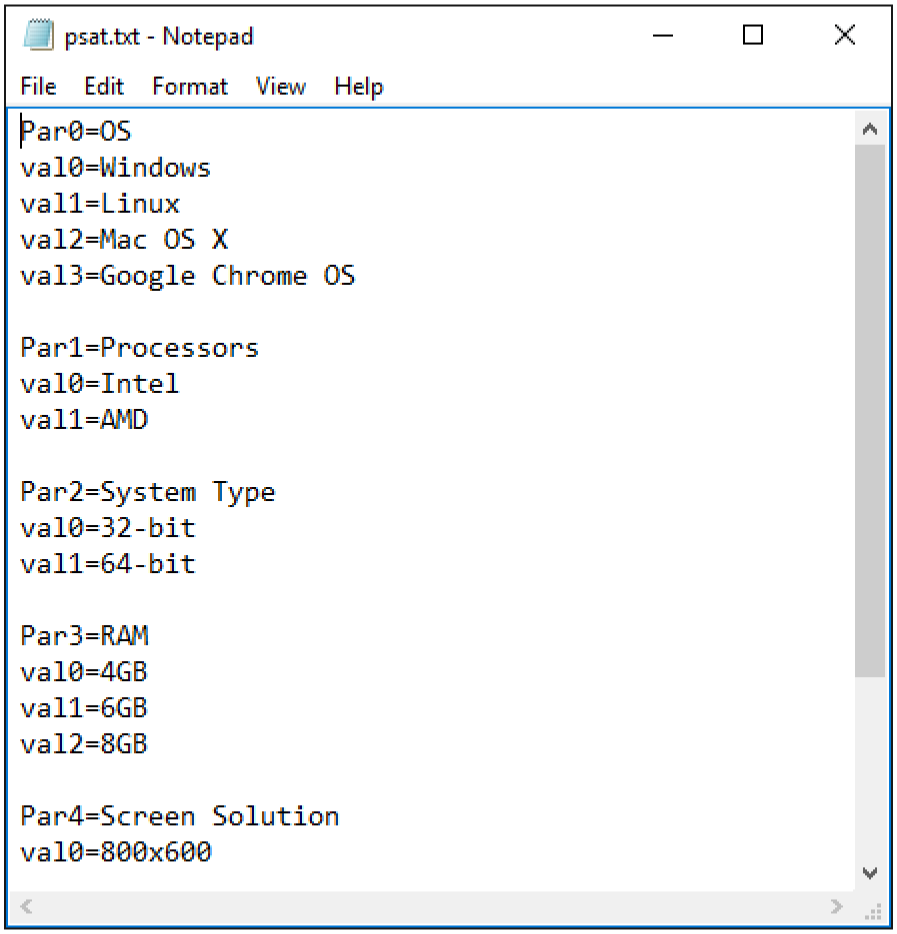
Firstly, the prototype interface will request some information from user such as number of input parameter, the name of each input parameter, number of values for each parameter and value names of each parameter. Figure 11 is the user interface of PSAT with designed on JFrame. The user is allowed to upload a file format to the user interface and edit the contents inside the file. Each time user editing the contents through the user interface, user requires saving the text file again before proceeds to the interaction list generation. Figure 12 is the file format to enter all the required input of this PSAT.


Universiti Malaysia PAHANG
Engineering • Technology • Creativity
FACULTY OF COMPUTER SYSTEMS & SOFTWARE ENGINEERING (FSKKP)
PSAT: A PAIRWISE TEST DATA GENERATION TOOL BASED ON SIMULATED ANNEALING ALGORITHM
Open File
Save File
Generate Test Case

```
Par0=OS
val0=Windows
val1=Linux
val2=Mac OS X
val3=Google Chrome OS

Par1=Processors
val0=Intel
val1=AMD

Par2=System Type
val0=32-bit
val1=64-bit

Par3=RAM
val0=4GB
val1=6GB
```

Copyrights © 2015 by Goh Ghee Hau (CB12079). All rights reserved.

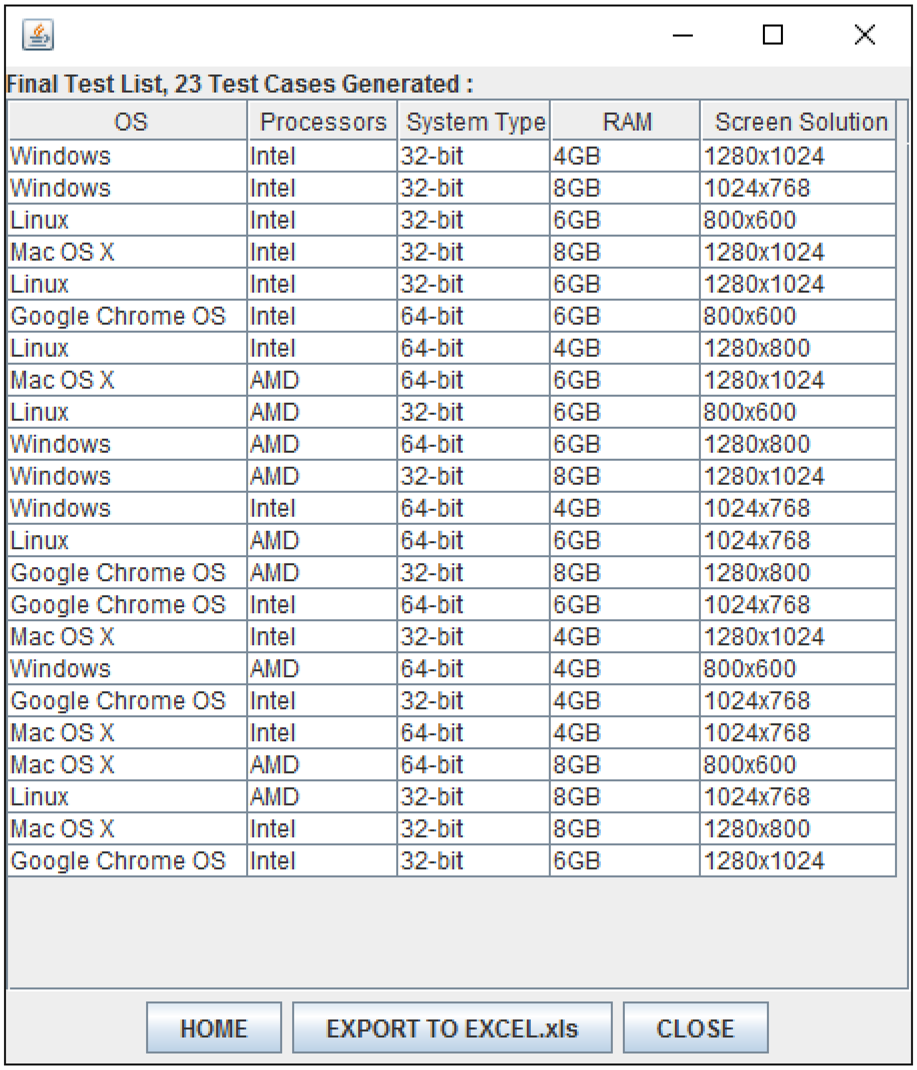
Figure 11 User Interface of PSAT



```
Par0=OS
val0=Windows
val1=Linux
val2=Mac OS X
val3=Google Chrome OS

Par1=Processors
val0=Intel
val1=AMD

Par2=System Type
val0=32-bit
val1=64-bit

Par3=RAM
val0=4GB
val1=6GB
val2=8GB

Par4=Screen Solution
val0=800x600
```

Figure 12 Text File Format for User Input

After user finish enter the require input, user need can generate the test case by pressing the Generate Test Case button on the user interface. The PSAT algorithm will be triggered and produce the actual result to user. Figure 13 is the final test cases with actual parameter name after completed the interaction list generation and test case generation.



Final Test List, 23 Test Cases Generated :

OS	Processors	System Type	RAM	Screen Solution
Windows	Intel	32-bit	4GB	1280x1024
Windows	Intel	32-bit	8GB	1024x768
Linux	Intel	32-bit	6GB	800x600
Mac OS X	Intel	32-bit	8GB	1280x1024
Linux	Intel	32-bit	6GB	1280x1024
Google Chrome OS	Intel	64-bit	6GB	800x600
Linux	Intel	64-bit	4GB	1280x800
Mac OS X	AMD	64-bit	6GB	1280x1024
Linux	AMD	32-bit	6GB	800x600
Windows	AMD	64-bit	6GB	1280x800
Windows	AMD	32-bit	8GB	1280x1024
Windows	Intel	64-bit	4GB	1024x768
Linux	AMD	64-bit	6GB	1024x768
Google Chrome OS	AMD	32-bit	8GB	1280x800
Google Chrome OS	Intel	64-bit	6GB	1024x768
Mac OS X	Intel	32-bit	4GB	1280x1024
Windows	AMD	64-bit	4GB	800x600
Google Chrome OS	Intel	32-bit	4GB	1024x768
Mac OS X	Intel	64-bit	4GB	1024x768
Mac OS X	AMD	64-bit	8GB	800x600
Linux	AMD	32-bit	8GB	1024x768
Mac OS X	Intel	32-bit	8GB	1280x800
Google Chrome OS	Intel	32-bit	6GB	1280x1024

Figure 13 Final Test Cases with Actual Parameter Name of PSAT

4.2.2 Interaction List Generation

Interaction list generation is the second technique in the development process of PSAT. The interaction list generation will be triggered after user enters the number of input parameter, name of each input parameter, number of values for each parameter and value names of each parameter then submitted it. Essentially, our proposed research is designed on the framework of pairwise testing, therefore the combinatorial interaction to generate the interaction list will be T=2 only.

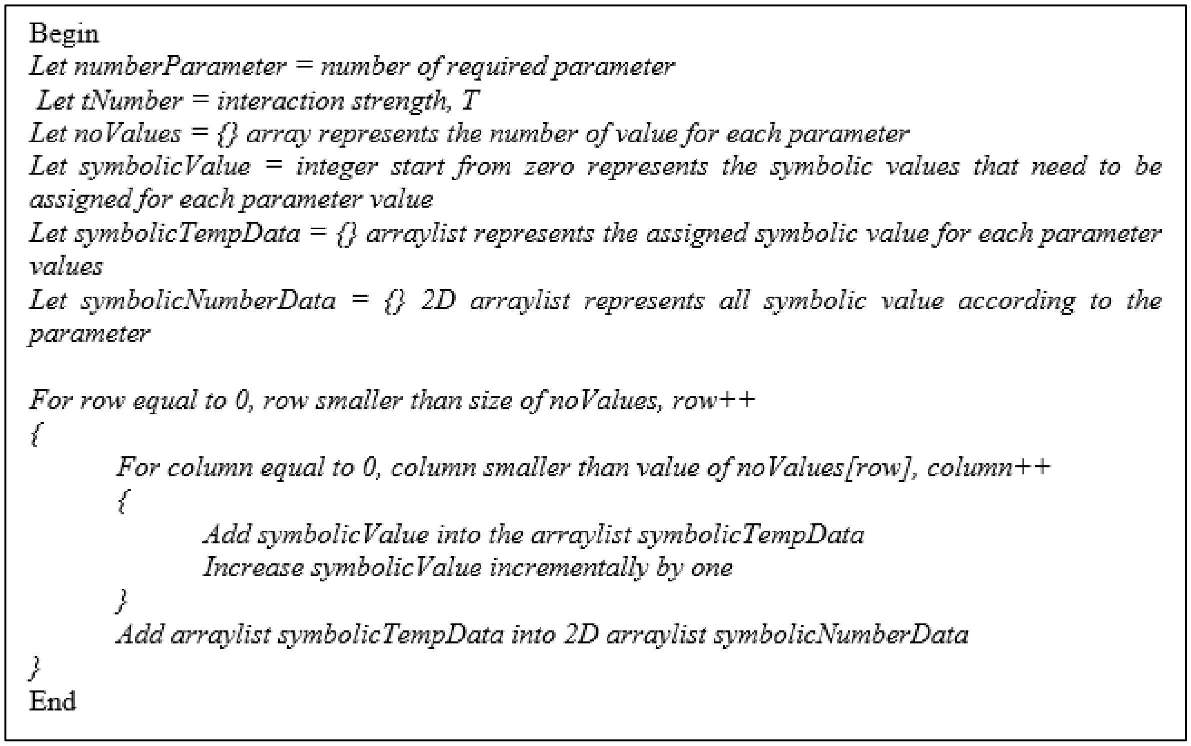
4.2.2.1 Automated Assigning for Parameter Values

In this section, an automated assigning method was implemented into this proposed research PSAT to ensure the program is faster processing and reduce the operation time. The automated assigning method is a process on converting the input parameter values into a set of integer number called symbolic values which are starting from 0, 1, 2, 3 and so on. In PSAT, each input parameter having different parameter values and names which enter by user. Therefore, the automated assigning will be triggered by assigning the symbolic values according to the total parameter values.

As shown in Table 13, the parameter values will be assigning the symbolic values starting from integer 0 until the parameter values is end. The symbolic values will be store in an arraylist and used for the generated interaction list.

Table 13 Assigning Symbolic Values for Specify Parameter Values

Parameter (P)	Values (V)	Symbolic Values (S)
OS	{Windows, Linux, Mac OS X, Google Chrome OS}	{0, 1, 2, 3}
Processors	{Intel, AMD}	{4, 5}
System Type	{32-bit, 64-bit}	{6, 7}
RAM	{4GB, 6GB, 8GB}	{8, 9, 10}
Screen Solution	{800×600, 1024×768, 1280×800, 1280×1024}	{11, 12, 13, 14}



```
Begin
Let numberParameter = number of required parameter
 Let tNumber = interaction strength, T
Let noValues = {} array represents the number of value for each parameter
Let symbolicValue = integer start from zero represents the symbolic values that need to be assigned for each parameter value
Let symbolicTempData = {} arraylist represents the assigned symbolic value for each parameter values
Let symbolicNumberData = {} 2D arraylist represents all symbolic value according to the parameter

For row equal to 0, row smaller than size of noValues, row++
{
        For column equal to 0, column smaller than value of noValues[row], column++
        {
                Add symbolicValue into the arraylist symbolicTempData
                Increase symbolicValue incrementally by one
        }
        Add arraylist symbolicTempData into 2D arraylist symbolicNumberData
}
End
```

Figure 14 Automated Assigning Algorithm for Each Parameter Values

4.2.2.2 Generating Interaction Pair

In PSAT, there is essential to generate the interaction pairs before generating the final test case. In pairwise testing, the interaction pairs of PSAT are generated based on the combination of every two parameters with their own values that have been specifying by the software tester. Table 14 shows the example of five (5) parameters with different values that have assigned with a symbolic value. P1 represents the first parameter, P2 represents the second parameter, P3 represents the third parameter and so on.

Table 14 Five Parameters with Symbolic Values

OS (P1)	Processors (P2)	System Type (P3)	RAM (P4)	Screen Solution (P5)
Windows (0)	Intel (4)	32-bit (6)	4GB (8)	800×600 (11)
Linux (1)	AMD (5)	64-bit (7)	6GB (9)	1024×768 (12)
Mac OS X (2)			8GB (10)	1280×800 (13)
Google Chrome OS (3)				1280×1024 (14)

As shown in Table 14, the possible interactions are P1P2, P1P3, P1P4, P1P5, P2P3, P2P4, P2P5, P3P4, P3P5 and P4P5 which are 10 possible interactions. Table 15 shows the possible interactions of the five parameters based on pairwise testing.

Table 15 Possible Interaction for P1, P2, P3, P4 and P5

No.	**P1**	**P2**	**P3**	**P4**	**P5**	**Possible Interactions**
T1	✓	✓				P1P2
T2	✓		✓			P1P3
T3	✓			✓		P1P4
T4	✓				✓	P1P5
T5		✓	✓			P2P3
T6		✓		✓		P2P4
T7		✓			✓	P2P5
T8			✓	✓		P3P4
T9			✓		✓	P3P5
T10				✓	✓	P4P5

With the possible interaction in Table 15, we generate these interactions based on binary number. Figure 15 shows the pseudocode for the generating possible interaction into binary number and save in interaction list based on interaction strength, T=2. We apply the use of pseudocode in Figure 15 to set the number of required parameter which is five for the example in Table 14 as *numberOfParameter* and the interaction strength, T=2 as *numberOfT* (Wen F. W. 2015). When the required parameter and interaction strength is accepted, the algorithm will be triggered and generate the binary number which consists of 0 and 1. Value of 0 is represents there is no interaction between the specify parameters while the value of 1 represents an existing interaction between the specify parameters.

```
Begin
Let numberOfParameter = number of required parameter
Let i = integer i
Let j = integer j
Let numberOfT = interaction strength, T
Let d = decimal

For i equal to (numberOfParameter – 1), i is equal or greater than 1, i--
{
        For j equal to (i – 1), j is equal or greater than 0, j--
        {
                $d = 2^i + 2^j$
                save d as interaction
        }
}
End
```

Figure 15 Pseudocode for Interaction List Generation When T=2

As an example, the binary number for T1 that show in Table 15 is 11000 which are generated from the solution of $2^4 + 2^3$. Each parameter will be assign one position values which 0, 1, 2, 3 and 4 to represent the position of the parameters. Then, the position values will use as the solution of base 2 which are $2^{[\text{the position values}]}$. Table 16 show the binary position for each possible interaction and Table 17 illustrate the position for each parameter.

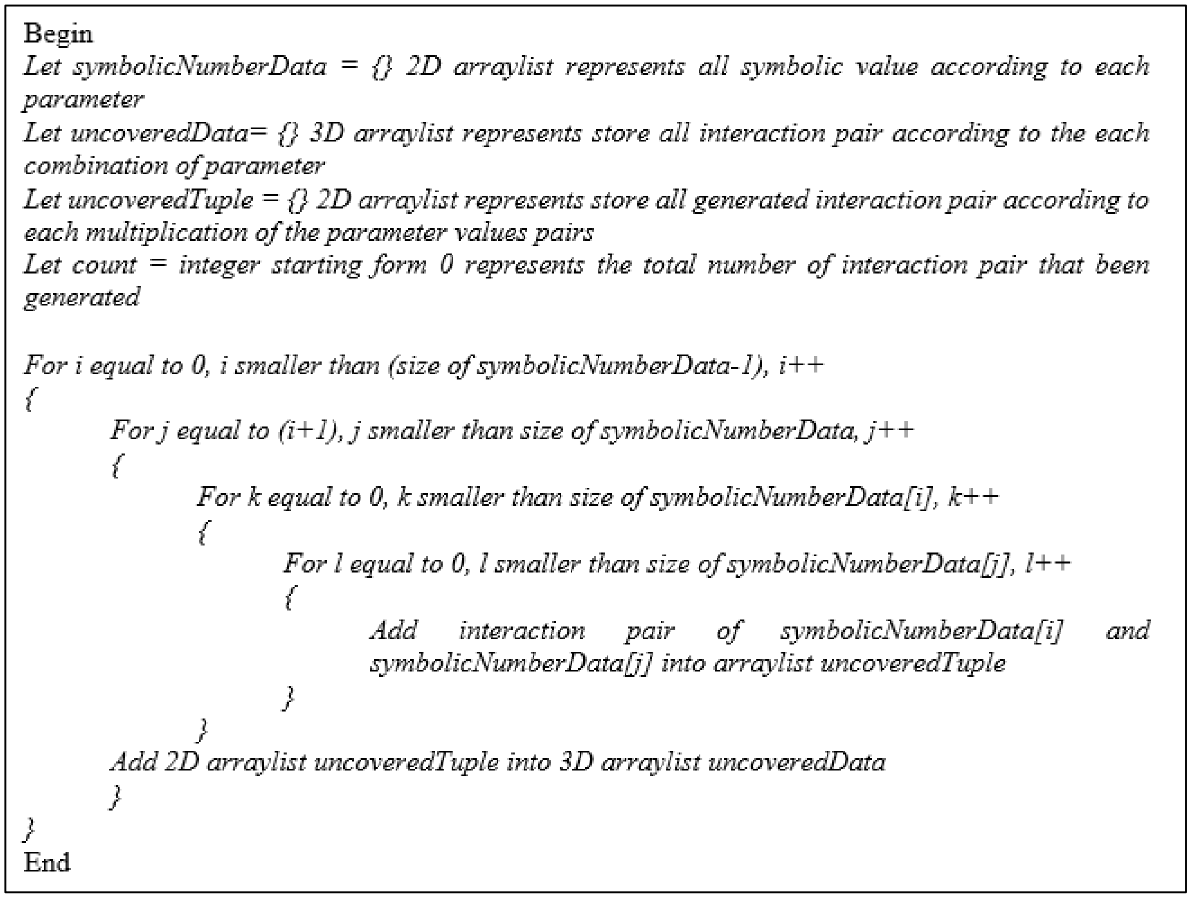
Table 16 Binary Position for Possible Interaction

No.	**P1**	**P2**	**P3**	**P4**	**P5**	**Interaction**	**Binary**
T1	✓	✓				P1P2	11000
T2	✓		✓			P1P3	10100
T3	✓			✓		P1P4	10010
T4	✓				✓	P1P5	10001
T5		✓	✓			P2P3	01100
T6		✓		✓		P2P4	01010
T7		✓			✓	P2P5	01001
T8			✓	✓		P3P4	00110
T9			✓		✓	P3P5	00101
T10				✓	✓	P4P5	00011

Table 17 Position of Each Parameter

Parameter	P1	P2	P3	P4	P5
Position Values	4	3	2	1	0
Solution of base 2	2^4	2^3	2^2	2^1	2^0

After finding the combination of parameter, the interaction list will be generated based on the two parameter values of the combination parameters interaction. Figure 16 shows the pseudocode of the generating interaction list.



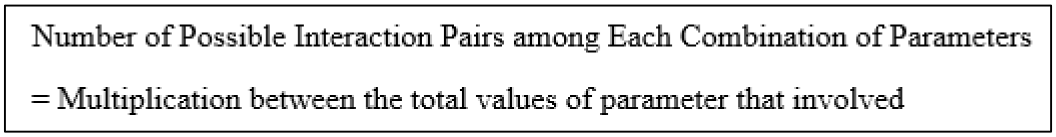
```
Begin
Let symbolicNumberData = {} 2D arraylist represents all symbolic value according to each parameter
Let uncoveredData= {} 3D arraylist represents store all interaction pair according to the each combination of parameter
Let uncoveredTuple = {} 2D arraylist represents store all generated interaction pair according to each multiplication of the parameter values pairs
Let count = integer starting form 0 represents the total number of interaction pair that been generated

For i equal to 0, i smaller than (size of symbolicNumberData-1), i++
{
        For j equal to (i+1), j smaller than size of symbolicNumberData, j++
        {
                For k equal to 0, k smaller than size of symbolicNumberData[i], k++
                {
                        For l equal to 0, l smaller than size of symbolicNumberData[j], l++
                        {
                                Add interaction pair of symbolicNumberData[i] and
                                symbolicNumberData[j] into arraylist uncoveredTuple
                        }
                }
        Add 2D arraylist uncoveredTuple into 3D arraylist uncoveredData
        }
}
End
```

Figure 16 Pseudocode of Interaction List Generation

As shown in Figure 16 above, the Interaction List Generation algorithm can be describes using the pseudocode above. Based on the Figure 16, the algorithm will trigger and started reading the *symbolicNumberData* which contains of all actual parameter values that been assigned with a symbolic values as we mentioned above.

In pairwise testing, the interaction strength is t=2, which is the generated interaction pair must be a combination of two parameter values only. So, the algorithm will read and loop in the *symbolicNumberData* to pair up the parameter values of each parameter according to the size. This means that when *symbolicNumberData[0]* represents the first parameter and *symbolicNumberData[1]* is represents the second parameter.

For more detail about the generation, here an example to discuss the generating interaction pair for every parameter. Based on the interaction in Table 15, the first interaction is P1P2 which parameter one (P1) consists of four (4) values with symbolic values {0, 1, 2, 3} and parameter two (P2) consists of two (2) values with symbolic values {4, 5}. Based on the values of each parameter, we can calculate the total possible interaction using the calculation as shown in Figure 17. For the first interaction pairs P1P2, the total possible interaction will be (4 x 2) = 8 pairs.

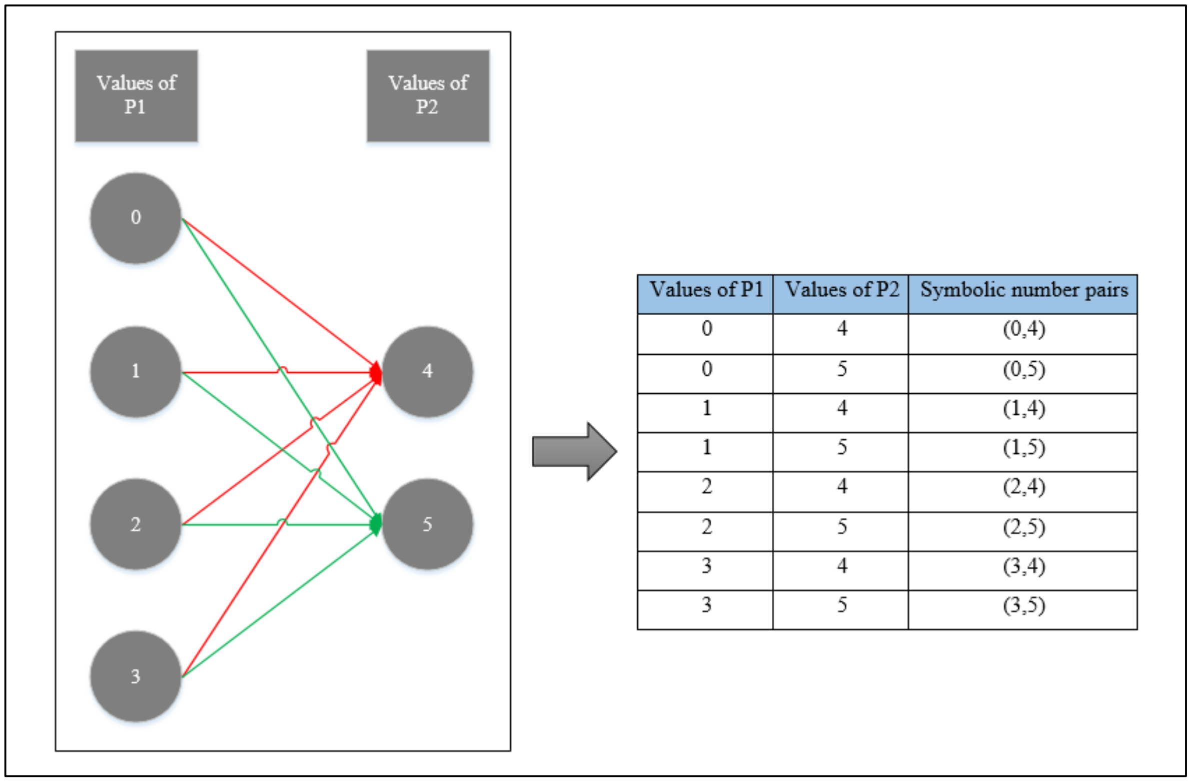


Number of Possible Interaction Pairs among Each Combination of Parameters

= Multiplication between the total values of parameter that involved

Figure 17 Calculation for Total Possible Interaction

Therefore, the possible interaction of the first interaction pairs P1P2 is {(0, 4), (0, 5), (1, 4), (1, 5), (2, 4), (2, 5), (3, 4), (3, 5)} which is total 8 pairs of interaction pairs. Figure 18 shows the structure for generating the interaction and the result of interaction list for P1P2 with the symbolic numbers.



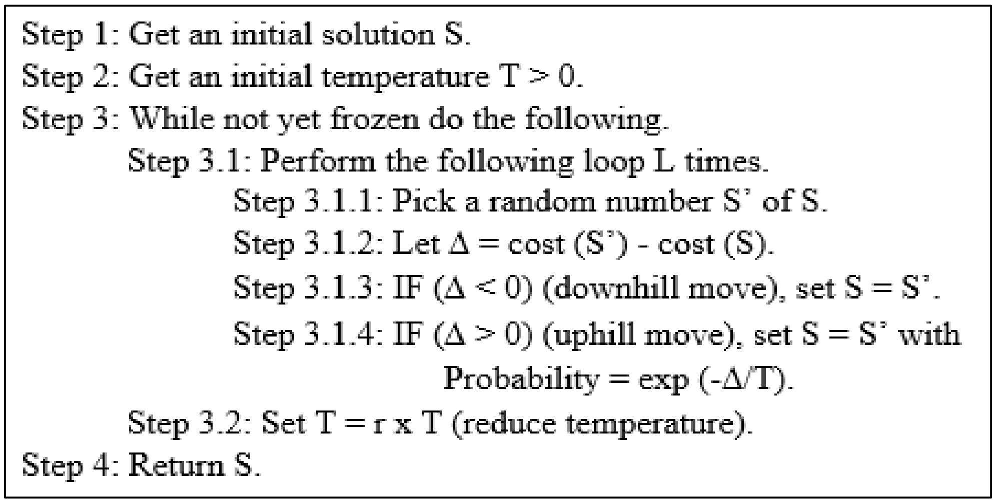
Values of P1	Values of P2	Symbolic number pairs
0	4	(0,4)
0	5	(0,5)
1	4	(1,4)
1	5	(1,5)
2	4	(2,4)
2	5	(2,5)
3	4	(3,4)
3	5	(3,5)

Figure 18 Structure Generating Interaction List with Symbolic Numbers

The algorithm will continue to generate all the possible interaction pairs with the remainder parameter which are P1P3, P1P4, P1P5, P2P3, P2P4, P2P5, P3P4, P3P5 and P4P5 before proceeding to the test case generation. All the possible interaction pairs will be saved in an arraylist and named *uncoveredData*.

4.2.3 Test Case Generation

Test Case Generation is the third technique in the development process of PSAT. In this proposed research, we will generate the test case using the Simulated Annealing (SA) algorithm. SA algorithm is a heuristic technique to find the feasible solution for the combinatorial optimization problems (Kirkpatrick S., Gelatt C. D. & Vecchi M. P. 1983). In order to understand more detail of our development, we will discuss the implementation of SA algorithm with several steps for this proposed research. In SA, the procedure is separate into four main steps as shown in Figure 19 (Kirkpatrick S., Gelatt C. D. & Vecchi M. P. 1983).



```
Step 1: Get an initial solution S.
Step 2: Get an initial temperature T > 0.
Step 3: While not yet frozen do the following.
        Step 3.1: Perform the following loop L times.
                Step 3.1.1: Pick a random number S' of S.
                Step 3.1.2: Let $\Delta$ = cost (S') - cost (S).
                Step 3.1.3: IF ($\Delta < 0$) (downhill move), set S = S'.
                Step 3.1.4: IF ($\Delta > 0$) (uphill move), set S = S' with
                                Probability = $\exp(-\Delta/T)$.
        Step 3.2: Set T = r x T (reduce temperature).
Step 4: Return S.
```

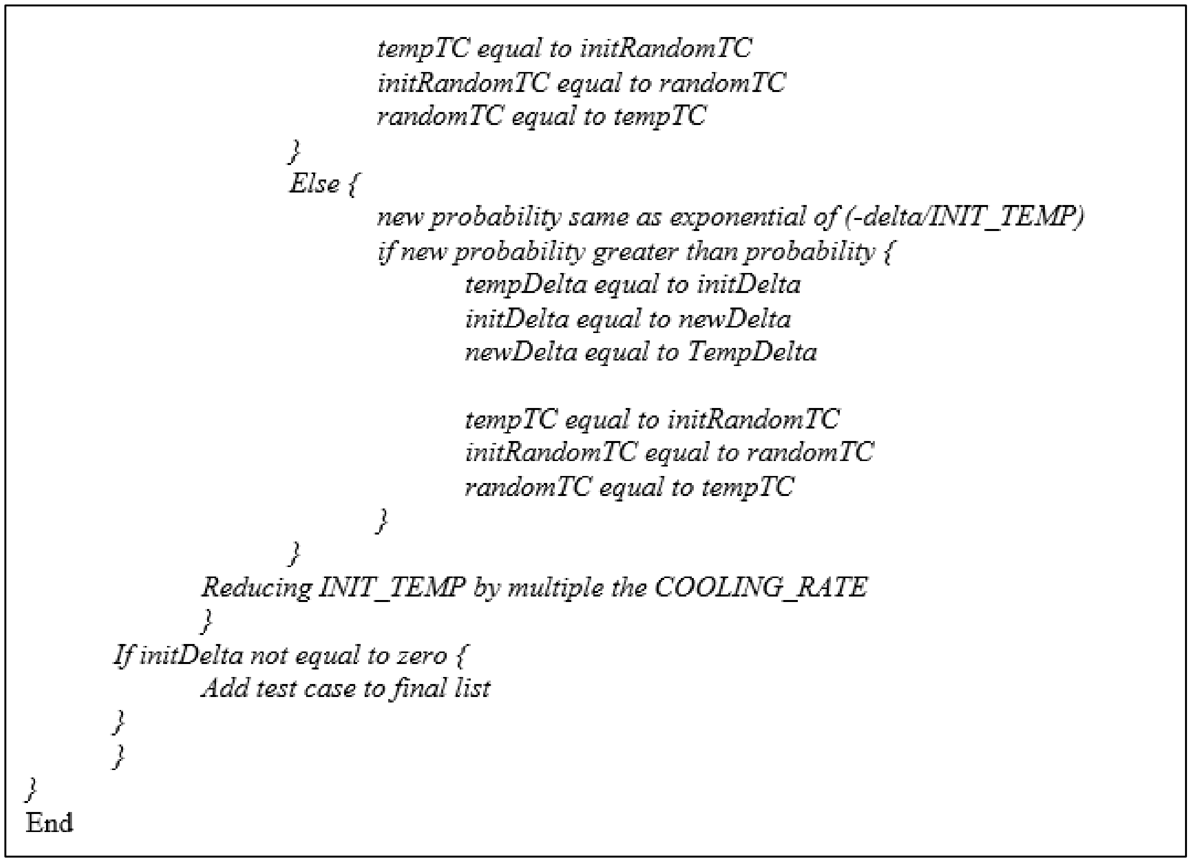
Figure 19 The SA Algorithm

Initially in Step 1, defining test case (S) becomes the initial test case that use to make comparison. In Step 2, defining an initial temperature (T) based on the input from software tester. This temperature (T) is used to check the movement whether goes uphill or downhill. Step 3 will continue to generate the test case to do comparison until the T become 0 or all the test case have covered. Each completion of a loop, the temperature is reducing to continue for the next loop. For each loop, we randomly pick one of the test case (S) as new test case (S'). In next step, we find out the difference between the new test case (S') and the initial test case (S). When the difference of the two test cases is less than zero, the downhill will be accepted and initial test case is remains. While the changes of two test cases are higher than zero, the uphill will be accepted and the test case needs to check based on the probability (P). If P is more than the difference of test case, the initial test case will be replaced by the new test case for the next generation. While P is less than the difference of test case, the initial test case is remaining for the next generation.

Based on the discussion above, we have implemented the SA algorithm with our own method into PSAT which is reducing the test case. PSAT Test Case Generation will be triggered after finish generating all the interaction pairs in Interaction List Generation. In this section, PSAT will generate all relevant test case based on SA algorithm. Figure 20a and Figure 20b is the pseudocode of the test case generation based on SA algorithm.

```
Begin
Let probability = represents the initial probability which is 0.7
Let COOLING_RATE = represents the initial cooling rate which is 0.3
Let tuplesCount = integer starting from 0 represents the total number of interaction pair that been
generated
Let maximumWeight = represents the total size of uncoveredData which given name tuples
initDelta = represents the weightage of the initial random test case
newDelta = represents the weightage of the random test case
Let INIT_TEMP = represents the initial temperature which is 10000
Let tempDelta = represents save delta as temporary delta
Let delta = represents the difference of newDelta and initDelta
Let tempTC = represents save test case as temporary test case
Let initRandomTC = represents initial test case from initial random test case
Let randomTC= represents test case from random test case

While tuplesCount no equal to zero {
        Generate initiate random test case
        If initDelta equal to maximumWeight {
                Add test case into final list
        }
        Else {
                While INIT_TEMP smaller than zero AND count no equal to zero {
                        Generate random test case
                        If delta equal or greater than zero {
                                tempDelta equal to initDelta
                                initDelta equal to newDelta
                                newDelta equal to TempDelta
```

Figure 20a Pseudocode of Test Case Generation Based on SA Algorithm



```
                tempTC equal to initRandomTC
                initRandomTC equal to randomTC
                randomTC equal to tempTC
            }
            Else {
                new probability same as exponential of (-delta/INIT_TEMP)
                if new probability greater than probability {
                    tempDelta equal to initDelta
                    initDelta equal to newDelta
                    newDelta equal to TempDelta

                    tempTC equal to initRandomTC
                    initRandomTC equal to randomTC
                    randomTC equal to tempTC
                }
            }
        Reducing INIT_TEMP by multiple the COOLING_RATE
        }
    If initDelta not equal to zero {
        Add test case to final list
    }
    }
}
End
```

Figure 20b Pseudocode of Test Case Generation Based on SA Algorithm

According to the Figure 20a and Figure 20b, the final test case will be generated by counting all the number of interaction pairs or tuples from *uncoveredData* and named it as *tuplesCount*. Every time a test case is adding into Final Test List, the interaction pair or the tuples that involved will be removed from and the *uncoveredData* and the *tuplesCount* will be count based on the current *uncoveredData*. When the *tuplesCount* is equal to zero, this means that all interaction pairs or tuples have been triggered and covered by all test cases in the Final Test List. Therefore, the PSAT execution will be terminating to generate new test case again.

Initially, the final test case is empty and *tuplesCount* is not empty, the PSAT algorithm will be executed and random generate test case. These random generate test case will be named as *initRandomTC* because the test case is covered the first pair interaction pairs or tuples inside the *uncoveredData*. Then the weightage of the test case will be calculated and check whether the weightage reached the maximum weightage. If the test case reaches the maximum weightage, the test case will be adding to the Final Test List. The interaction pairs or tuples will be removed from *uncoveredData* and *tuplesCount* will be decrease also.

If the test case does not cover the maximum weightage, the PSAT algorithm with set initial temperature and loop to generate random test case named as *randomTC* until the temperature become 0.0. In this step, the weightage of the *initRandomTC* and *randomTC* will be calculated and find the difference between these two test case. The difference weightage of the two test cases named delta (*newDelta* - *initDelta*). When delta equals or greater than zero, the two weightage and two test cases will be exchanges. Then *newDelta* and new *randomTC* will be used to compare with the new random generation test case again by reducing the temperature each time. When the temperature become 0.0, last generated test case will be added to the Final Test List and removed the tuples that involved from the *uncoveredData* if the *initDelta* is not equal to zero.

There is one condition when the delta not equal or greater than zero, here we need to check the probability of the two test case and find out the suitable test case to use for the next random generation test case. In this step, we initially the probability equal to 0.7 which means the new probability having 70% better than the initial probability, the new test case will be accepted to use for the next random generation test case else the test case is remains. This is same as previous, the two weightage and two test cases with be exchange also and used to compare with the new random generation test case again by reducing the temperature. When temperature is equal to 0.0 and *initDelta* not equal to zero, the last generated test case will be added to the Final Test List and removed the tuples that involved from the *uncoveredData* again. Figure 21 is the flowchart of PSAT algorithm based on SA algorithm.

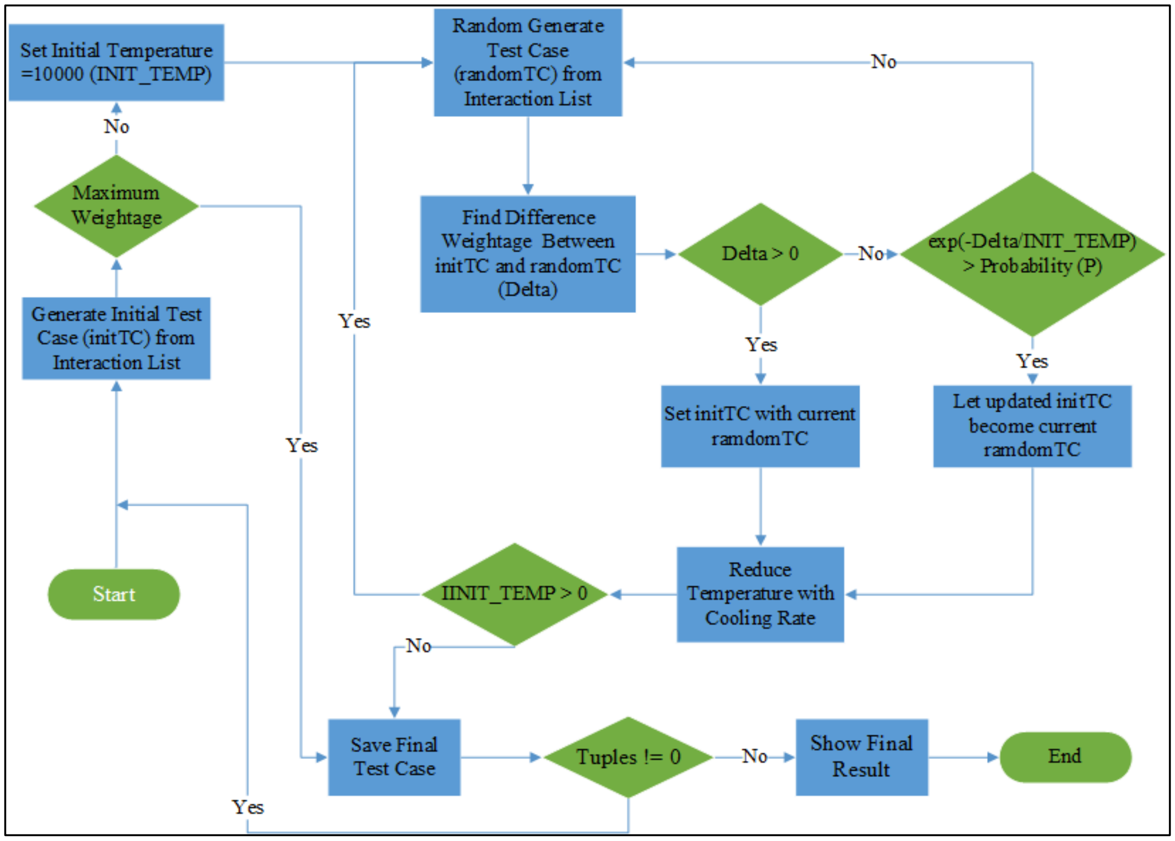
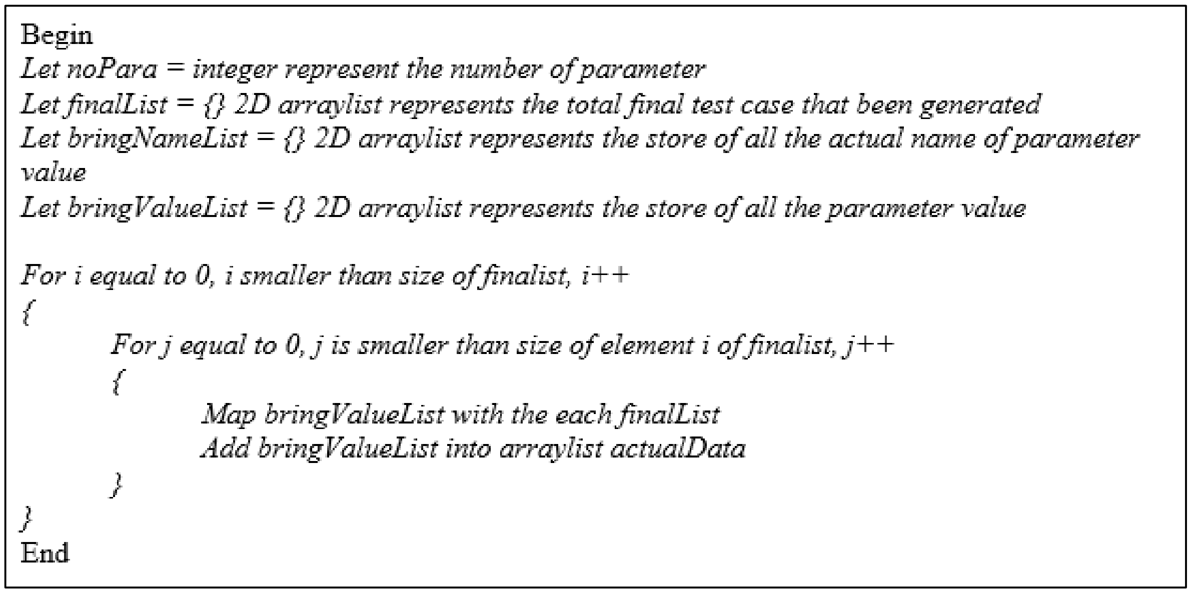

Set Initial Temperature =10000 (INIT_TEMP)
Random Generate Test Case (randomTC) from Interaction List
No
No
Maximum Weightage
Find Difference Weightage Between initTC and randomTC (Delta)
Delta > 0
No
exp(-Delta/INIT_TEMP) > Probability (P)
Generate Initial Test Case (initTC) from Interaction List
Yes
Yes
Yes
Set initTC with current ramdomTC
Let updated initTC become current ramdomTC
Yes
Start
IINIT_TEMP > 0
Reduce Temperature with Cooling Rate
No
Save Final Test Case
Tuples != 0
No
Show Final Result
End
Yes

Figure 21 Flow Chart of PSAT Algorithm

4.2.4 Mapping Generation Algorithm

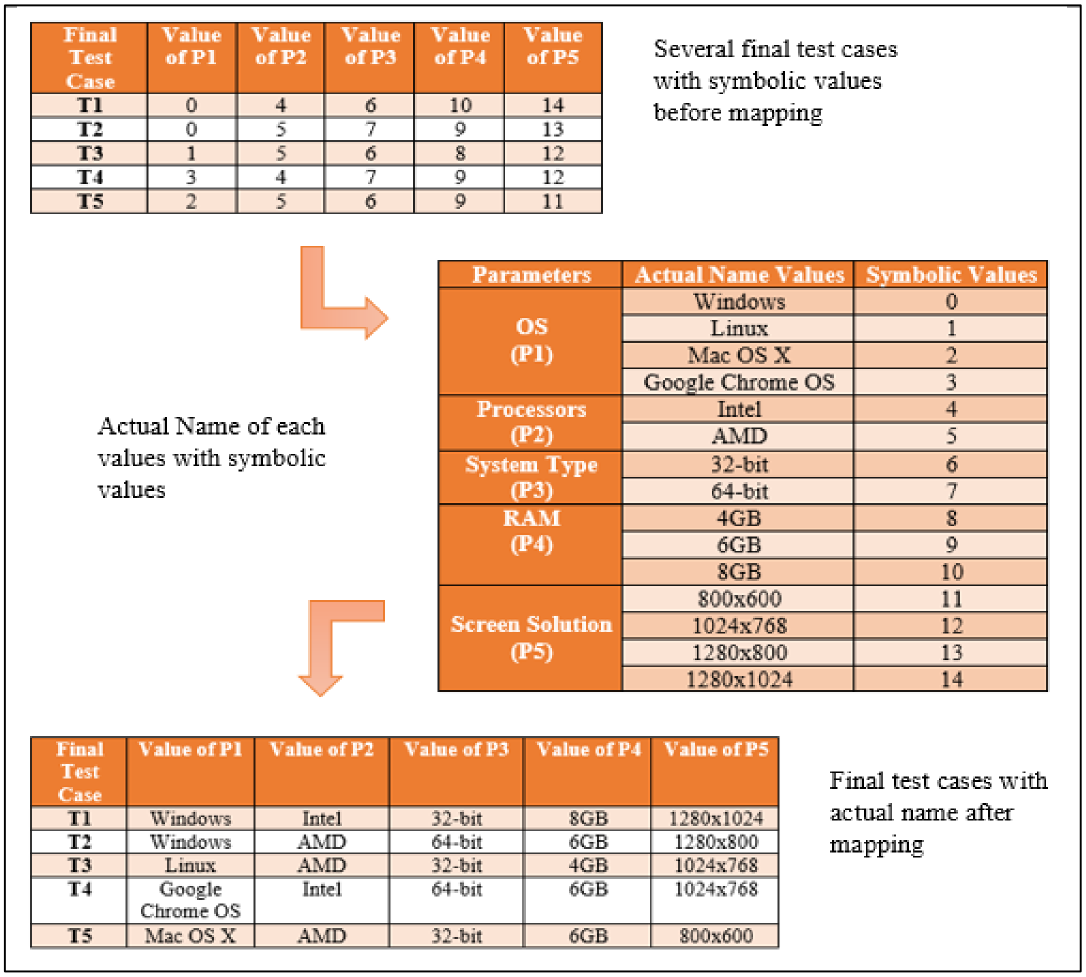
In previous development, PSAT has converted the actual parameter values name into symbolic values to generate final test cases. This is the four techniques which are to let the whole PSAT become faster and reduce time. Therefore, after the final test case has been generated, the symbolic values should be converted back to the actual parameter values name to make user understand the actual final test cases. To success the process, Mapping Generation algorithm has applied in PSAT to map the symbolic values back to the actual parameter values name. Figure 22 show the Mapping Generation algorithm used for PSAT.



```
Begin
Let noPara = integer represent the number of parameter
Let finalList = {} 2D arraylist represents the total final test case that been generated
Let bringNameList = {} 2D arraylist represents the store of all the actual name of parameter
value
Let bringValueList = {} 2D arraylist represents the store of all the parameter value

For i equal to 0, i smaller than size of finalist, i++
{
        For j equal to 0, j is smaller than size of element i of finalist, j++
        {
                Map bringValueList with the each finalList
                Add bringValueList into arraylist actualData
        }
}
End
```

Figure 22 Mapping Generation Algorithm

After all the final test cases are generated, the Mapping Generation algorithm will be triggered and read the size of the *finalList*. Then, map the symbolic values of each final test case with the actual parameter values name which is named *bringValueList*. After mapping, save the actual name test case in another arraylist. This process will be repeated until the *finalList* is finished and all of the symbolic values test case should success converted back to the actual parameter values name. Figure 23 the mapping results of the symbolic values and parameter values.



Final Test Case	Value of P1	Value of P2	Value of P3	Value of P4	Value of P5
T1	0	4	6	10	14
T2	0	5	7	9	13
T3	1	5	6	8	12
T4	3	4	7	9	12
T5	2	5	6	9	11

Several final test cases with symbolic values before mapping

Parameters	Actual Name Values	Symbolic Values
OS (P1)	Windows	0
	Linux	1
	Mac OS X	2
	Google Chrome OS	3
Processors (P2)	Intel	4
	AMD	5
System Type (P3)	32-bit	6
	64-bit	7
RAM (P4)	4GB	8
	6GB	9
	8GB	10
Screen Solution (P5)	800x600	11
	1024x768	12
	1280x800	13
	1280x1024	14

Actual Name of each values with symbolic values

Final Test Case	Value of P1	Value of P2	Value of P3	Value of P4	Value of P5
T1	Windows	Intel	32-bit	8GB	1280x1024
T2	Windows	AMD	64-bit	6GB	1280x800
T3	Linux	AMD	32-bit	4GB	1024x768
T4	Google Chrome OS	Intel	64-bit	6GB	1024x768
T5	Mac OS X	AMD	32-bit	6GB	800x600

Final test cases with actual name after mapping

Figure 23 Mapping Results of the Symbolic Values and Parameter Values

4.3 Result and Discussion

After completing the implementation of PSAT, PSAT is evaluated and analyzes to ensure the goals and objectives are achieved. In this proposed research, the main objective is to evaluate the PSAT in term of test size which based on pairwise testing, t=2. Therefore, PSAT will be evaluated and compare with the existing result which collected from other existing pairwise testing strategies that mentioned in the literature review.

There is two main existing results that collected from the previous literature review which shown in Table 19 and Table 22. We separate the two results into two experiments by given named Experiment 1 (E1) and Experiment 2 (E2). To conduct this experiment, the test case will be generated by using an Asus K43SD laptop with Windows 10 Education and processors Intel® Care™ i5-2450M CPU @ 2.50 GHz, 6GB DDR3 SSDRAM and 2.5" SATA 500GB hard disk. NetBeans 8.0.2, Oracle JDK 8.0 and JRE 8.0 are the application that needs to be installed to conduct this experiment also.

Experiment 1

In the Experiment 1, the test case is generated based on the interaction strength, t=2. Each experiment result is running for 10 times to get the average results of the test cases and the average of execution time. The existing pairwise testing strategies tools are used for comparing with the PSAT are SA_SAT, mAETG_SAT (Alsewari, A. R. A. & Zamli, K. Z. 2012), PICT (Ahmed. B. S. & Zamli, K. Z. 2011), TestCover, LAHC, HSS and BA_PTCS (Alsariera, Y. A., Alsewari, A. R. A. & Zamli, K. Z. 2015). In Experiment 1, there are five (5) tests are run with the different input specifications as shown in Table 18.

Table 18 Input Specifications for Experiment 1

Tests	Covering Array Representation	Input Specification
S1	CA(N,2,3^3)	three 3-valued parameters
S2	CA(N,2,4^3)	three 4-valued parameters
S3	CA(N,2,5^3)	three 5-valued parameters
S4	CA(N,2,6^3)	three 6-valued parameters
S5	CA(N,2,7^3)	three 7-valued parameters

Table 19 Results for Other Existing Pairwise Strategies Tools in Experiment 1

E1	SA_SAT	mAETG_SAT	PICT	TestCover	LAHC	HSS	BA-PTCS	PSAT
S1	9	9	10	9	9	9	9	11
S2	16	16	17	16	16	16	16	20
S3	25	25	26	25	25	25	25	36
S4	36	37	39	36	38	36	36	53
S5	49	52	55	49	51	49	49	77

Table 20 Results for PSAT in Experiment 1

	Test Case Generated		Time Execution (seconds)	
E1	PSAT (Minimum)	PSAT (Average)	PSAT (Minimum)	PSAT (Average)
S1	11	13.0	7	7.7
S2	20	24.8	8	8.4
S3	36	41.2	7	7.9
S4	53	58.1	7	7.9
S5	77	82.5	8	8.2

Table 19 is the existing results for several existing pairwise strategies tools used to compare with our PSAT and Table 19 is the overall results generated that carry out from Experiment 1. As shown in Table 20, the goals and objectives of PSAT have been achieved due to the results provided show that PSAT is reducing the test cases. For example, the Test S1 having three 3-valued parameters, which will produce 3 x 3 x 3 = 27 test cases. But using PSAT, the final test case has been reducing from 27 test cases to an average of 13 test cases.

By comparing with other existing results shown in Table 19, the PSAT result is not showing the best result between the other existing results after 10 times running. As an example in Test S2, PSAT has generated 20 test cases which are the minimum test case for the 10 times running with the same input specifications in Experiment 1. The result is a bit higher than other existing pairwise strategies result such as 17 test cases generated in PICT because PSAT is using a random method to generate all the test cases by reducing the temperature in each time. For generated a test case in PSAT, the total average of generation is about 24.8 test cases each time for Test S2.

Although the result of the PSAT is not the best, but the execution time to generate final test cases showing high efficiency. In Table 20, Test S2 shows that the minimum execution time of PSAT to generate all the final test case is 8 seconds only and the execution is very fast in generation test cases.

Experiment 2

In Experiment 2, the test case is generated based on the interaction strength, t=2. Each experiment result is running for 10 times to get the average results of the test cases and the average of the execution time. The existing pairwise testing strategies tools used for comparing the PSAT are AETG (Cohen, D. M., Dalal, S. R., Fredman, M. L. & Patton, G. C. 1997), AETG2 (Shiba, T., Tsuchiya, T. & Kikuno, T. 2004), IPO (Lei, Y. &Tai, K. C. 1998), SA, GA, ACA (Shiba, T., Tsuchiya, T. & Kikuno, T. 2004), ALL-Pairs, G2Way, Jenny and IPRS (Younis, M. I., Zamli, K. Z. & Isa, N. A. M. 2008). In Experiment 2, there are seven (7) tests are run with the different input specifications as shown in Table 21.

Table 21 Input Specifications for Experiment 2

Tests	Covering Array Representation	Input Specification
S1	CA(N,2,3^3)	three 3-valued parameters
S2	CA(N,2,3^4)	four 3-valued parameters
S3	CA(N,2,3^{13})	13 3-valued parameters
S4	CA(N,2,10^{10})	10 10-valued parameters
S5	CA(N,2,15^{10})	10 5-valued parameters
S6	CA(N,2,10^{20})	20 10-valued parameters
S7	CA(N,2,5^{10})	10 5-valued parameters

Table 22 Results for Other Existing Pairwise Strategies Tools in Experiment 2

E2	AETG	AETG2	IPO	SA	GA	ACA	ALL-Pairs	G2Way	Jenny	IPRS	PSAT
S1	NA	NA	NA	NA	NA	NA	10	10	9	9	12
S2	9	11	9	9	9	9	10	10	13	9	16
S3	15	17	17	16	17	17	22	19	20	17	34
S4	NA	NA	169	NA	157	159	177	160	157	149	419
S5	NA	NA	361	NA	NA	NA	390	343	336	321	958
S6	180	198	212	183	227	225	230	200	194	210	560
S7	NA	NA	47	NA	NA	NA	49	46	45	45	98

Table 23 Results for PSAT in Experiment 2

	Test Case Generated		Time Execution (seconds)	
E1	PSAT (Minimum)	PSAT (Average)	PSAT (Minimum)	PSAT (Average)
S1	12	13.2	8	9.5
S2	16	18.3	8	8.9
S3	34	39.4	8	9.1
S4	419	427.9	13	13.5
S5	958	974.9	27	27.7
S6	560	577.3	27	29.0
S7	98	106.4	8	9.1

Same as the Experiment 1, Table 21 is the existing results for several existing pairwise strategies tools used to compare with our PSAT and Table 22 is the overall results generated that carry out from Experiment 2.

Based on the result in Table 22, PSAT is successful to reduce the exhaustive test cases but does not show the best result comparing with the result of other existing pairwise strategies tools. As an example, PSAT has generated 34 test cases as minimum test cases and 39.4 test cases as the average test cases in Test S3. Most of the result from other existing pairwise strategies tools might better than PSAT which some existing pairwise strategies tool like AETG has generated 15 test cases only.

In term of the time execution, PSAT still considers as the faster tools to generate the final test cases. For an example, PSAT has generated an average of 974.9 test cases for Test S5. Normally, the execution time of generating the huge test cases will take a long time to execute, but the result shows that PSAT using an average 27.7 seconds to generate final test cases in 10 times running of the same input specifications.

Besides that, PSAT able to support all the different input specification in Table 20. As an evidence, certain tests like Test S1 which are three 3-valued parameter in Experiment 2 is not supported by other existing tools such as AETG, AETG2, IPO, SA, GA and ACA. Therefore, PSAT might better than other existing tools in term of generating test cases based on different input specification.

4.4 Summary

This chapter discusses the design and implementation of PSAT. The development process of PSAT is separate in four main techniques which included user interface, interaction list generation, test case generation and mapping generation. Input parameters and parameter values from the user will be represented by using integer number named symbolic values to faster the processing and reduce the operation time. Besides that, SA algorithm is discussed and applied into PSAT to generate the final test cases. After that, the mapping generation is used to map the final test cases to actual name to show user.

Lastly, PSAT has been analyzed and evaluated that implementation of PSAT able to generate test case by covered all the interaction pairs based on SA algorithm. PSAT also used in the experiment to compare with the other pairwise testing existing tools in term of test cases size. From the results, PSAT able to reduce the test cases size but not the best compare with other. But the results show that PSAT able to generate test cases based on different input specification and the execution time to finish generation is shorter.

CHAPTER 5

CONCLUSION

5.1 Introduction

In all previous chapters, we have discussed the flow and process to complete this research such as reviewing the existing pairwise testing strategies, the selected methodology, design and development of PSAT and also comparison results and discussion between PSAT and other pairwise testing tools. Last but not least, this chapter is concluded the research and future work for PSAT.

Based on the first chapter, the main goal of this research is to develop PSAT: A Pairwise Test Data Generation Tool Based on Simulated Annealing (SA) algorithm which able to generate sufficient test cases. The PSAT tool was developed using the NetBeans 8.0.2 application and Java Languages to provide an independent platform for all user to use it. To reaches the achievement of goals, following objectives have been set:

i) To study the pairwise testing generation by reviewing the existing pairwise testing strategies.
ii) To apply Simulated Annealing (SA) algorithm into PSAT.
iii) To evaluate and compare the performance of proposed PSAT against with other existing strategies in term of test size.

In order to achieve the goals and objectives, there are many aspects have been considered during the development of PSAT. In Chapter 2, there is necessary to review several existing pairwise testing strategies to gain knowledge and understanding for the development of PSAT. The PSAT is designed based on four main techniques which are user interface, interaction list generation, test case generation and mapping generation.

Furthermore, some experiment has been carried out to evaluate and prove the effectiveness of PSAT in Chapter 4. After the experiment, the result proves that PSAT able to reduce the exhaustive test cases with covered all the interaction pairs. Besides that, PSAT also proves that the execution time is faster by running with different input specification.

With the experiment results in this research, the performance of PSAT successfully reached the goals and objectives in term of reducing the test cases size. Although the PSAT not the best result during the comparison with other existing tools, but the execution time was the benefit of PSAT.

5.2 Research Constraint

To completion every research or project, there are several constraints of the research or project. This is same as completion of PSAT, following are the constraints of this research:

i) Limitation of time

During the Final Year Project, the result of PSAT does not show the best because we faced the limitation of time to run all experiment again and again until getting the best result of the different input specification. So, the result of PSAT is running in 10 times only.

ii) Limitation of test cases

As mentioned above, PSAT still have some limitation of generating minimum size of final test cases compare with other existing pairwise testing strategies. The reason for this constraint is because the SA algorithm having many random methods and each test case generated will reduce by temperature.

iii) Limitation of choosing local variables

Due to the limitation of time as mentioned above, we face another constraint which is the limitation of choosing local variables. In PSAT algorithm, there are many local variables such as temperature (0 to infinite), cooling rate, and probability is having a range of number from zero (0) to one (1). If we adjust the values of the local variables, it will take a long time to run all the experiment, so we set the temperature at 10000, the cooling rate is set on 0.3, and the probability is set on 0.7.

5.3 Future Work

For future enhancement of PSAT, there are several area of research can be carried out:

i) Improve PSAT algorithm

Based on the result, PSAT still has the limitation of generating minimum size of test cases. In the future improvements, PSAT algorithm should be modified and improve the strategy to generate optimal test cases.

ii) Support t-way interaction

For current research, PSAT only support pairwise interaction which is t=2. For future improvements, PSAT should be supported more than 2-way interaction ($t > 2$) to let the tool more flexible in future.

iii) Support constraint

In the real environment or system, some input parameter values should be included or excluded. Therefore, in future improvements, PSAT should be developed with support constraint to improve the practicability of test case generation.

iv) Support test process

In the Software Development Life Cycle, the test process included planning, analysis and design, implementation, testing, evaluating exit criteria and reporting, and test closure activities. So, PSAT should integrate with test process to be useful in the process of testing.

REFERENCES

Ahmed. B. S. & Zamli, K. Z. (2011). A Variable Strength Interaction Test Suites Generation Strategy Using Particle Swarm Optimization. *Journal of Systems and Software, vol. 84, 12, 2011, pp. 2171-2185*.

Alsariera, Y. A., Alsewari, A. R. A. & Zamli, K. Z. (2015). A Bat-Inspired Strategy for Pairwise Testing with Constraints Support. *Advanced Science Letters, Vol.21,8,2281-2284, 2015*.

Alsewari, A. R. A., & Zamli, K. Z. (2012). Design and implementation of a harmony-search-based variable-strength t-way testing strategy with constraints support. *Information and Software Technology, 54*(6), 553-568. doi: 10.1016/j.infsof.2012.01.002

Cohen, D. M., Dalal, S. R., Fredman, M. L. & Patton, G. C. (1997). The AETG System An Approach to Testing. *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 23, NO. 7, JULY 1997*.

Flores, P. & Cheon, Y. (2011). PWiseGen: Generating Test Cases for Pairwise Testing Using Genetic Algorithms. *IEEE, 978-1-4244-8728-8/11*.

Grindal, M., Offutt, J. & Andler, F. (2004). Combination Testing Strategies: A Survey.

Kirkpatrick S., Gelatt C. D. & Vecchi M. P. (1983). Optimization by Simulated Annealing. *Science, New Series, Vol. 220, No. 4598.*, 671-680.

Lei, Y., Kacker, R., Kuhn, D. R., Okun, V., & Lawrence, J. (2007). IPOG A General Strategy for T-Way Software Testing. *14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems*, 549-556.

Lei, Y., Kacker, R., Kuhn, D. R., Okun, V., & Lawrence, J. (2008). IPOG/IPOG-D: efficient test generation for multi-way combinatorial testing. *Software Testing, Verification and Reliability, 18*(3), 125-148. doi: 10.1002/stvr.381

Lei, Y. & Tai, K. C. (1998). In-Parameter-Order A Test Generation Strategy for Pairwise Testing. *Department of Computer Science, North Carolina State University, Raleigh, NC 27695-7534,USA.*

McCaffrey, J. D. (2009). Generation of Pairwise Test Sets Using a Genetic Algorithm. *33rd Annual IEEE International Computer Software and Applications Conference*, 626-631. doi: 10.1109/compsac.2009.91

McCaffrey, J. D. (2010). An Empirical Study of Pairwise Test Set Generation Using a Genetic Algorithm. *Seventh International Conference on Information Technology*, 992-997. doi: 10.1109/itng.2010.93

Mitchell, M. (1995). Genetic Algorithms: An Overview. *Complexity*, (1 (1)), 31-39.

Rutenbar R. A. (1989). Simulated Annealing Algorithms: An Overview. *IEEE 8755-3996/89/0100-0019*.

Myers, G. J., Badgett, T., & Sandler, C. (1979). The Art of Software Testing (3rd ed.). *JohnWiley & Sons, Inc., Hoboken, New Jersey.*

Patil, M., & Nikumbh, P. J. (2012). Pair-wise Testing Using Simulated Annealing. *Procedia Technology, 4*, 778-782. doi: 10.1016/j.protcy.2012.05.127

Perrouin, G., Oster, S., Sen, S., Klein, J., Baudry, B., & le Traon, Y. (2011). Pairwise testing for software product lines: comparison of two approaches. *Software Quality Journal, 20*(3-4), 605-643. doi: 10.1007/s11219-011-9160-9

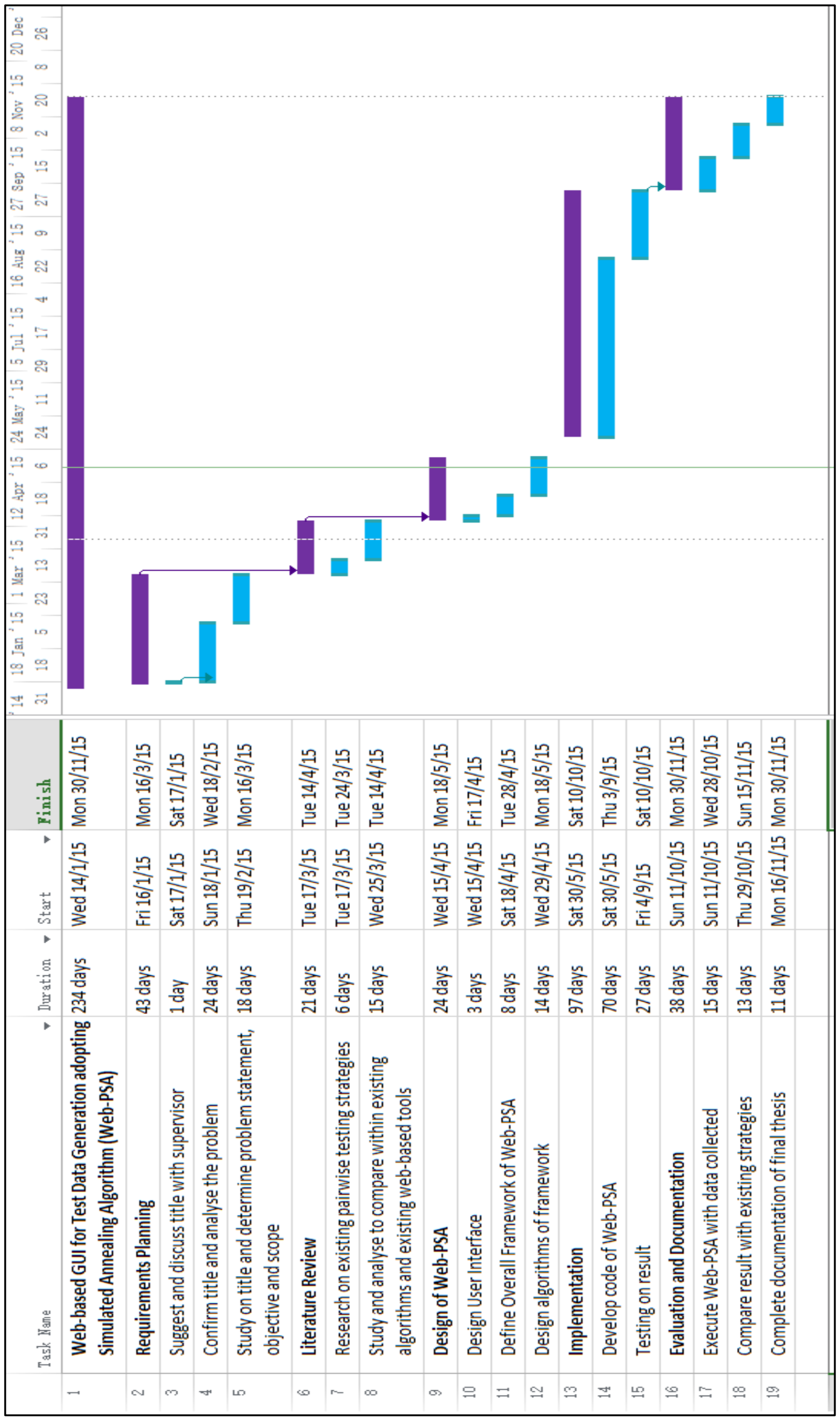
Shiba, T., Tsuchiya, T., Kikuno, T. (2014). Using Artificial Life Techniques to Generate Test Cases for Combinatorial Testing. *28th Annual International Computer Software and Applications Conference (COMPSAC'04), 0730-3157/04*.

Wang, S., Ali, S., & Gotlieb, A. (2013). Minimizing Test Suites in Software Product Lines Using Weight-based Genetic Algorithms. *Simula Research Laboratory, Technical Report 2012*, 25.

Younis, M. I., Zamli, K. Z. & Isa, N. A. M. (2008). IRPS - An Efficient Test Data Generation Strategy for Pairwise Testing.

Younis, M. I., Zamli, K. Z., Klaib, M. F. J., Soh, Z. H. C., Abdullah, S. A. C., & Isa, N. A. M. (2010). Assessing IRPS as an efficient pairwise test data generation strategy. *Int. J. Advanced Intelligence Paradigms, Vol. 2, No. 1, 2010*.

APPENDICES



	Task Name	Duration	Start	Finish
1	**Web-based GUI for Test Data Generation adopting Simulated Annealing Algorithm (Web-PSA)**	234 days	Wed 14/1/15	Mon 30/11/15
2	**Requirements Planning**	43 days	Fri 16/1/15	Mon 16/3/15
3	Suggest and discuss title with supervisor	1 day	Sat 17/1/15	Sat 17/1/15
4	Confirm title and analyse the problem	24 days	Sun 18/1/15	Wed 18/2/15
5	Study on title and determine problem statement, objective and scope	18 days	Thu 19/2/15	Mon 16/3/15
6	**Literature Review**	21 days	Tue 17/3/15	Tue 14/4/15
7	Research on existing pairwise testing strategies	6 days	Tue 17/3/15	Tue 24/3/15
8	Study and analyse to compare within existing algorithms and existing web-based tools	15 days	Wed 25/3/15	Tue 14/4/15
9	**Design of Web-PSA**	24 days	Wed 15/4/15	Mon 18/5/15
10	Design User Interface	3 days	Wed 15/4/15	Fri 17/4/15
11	Define Overall Framework of Web-PSA	8 days	Sat 18/4/15	Tue 28/4/15
12	Design algorithms of framework	14 days	Wed 29/4/15	Mon 18/5/15
13	**Implementation**	97 days	Sat 30/5/15	Sat 10/10/15
14	Develop code of Web-PSA	70 days	Sat 30/5/15	Thu 3/9/15
15	Testing on result	27 days	Fri 4/9/15	Sat 10/10/15
16	**Evaluation and Documentation**	38 days	Sun 11/10/15	Mon 30/11/15
17	Execute Web-PSA with data collected	15 days	Sun 11/10/15	Wed 28/10/15
18	Compare result with existing strategies	13 days	Thu 29/10/15	Sun 15/11/15
19	Complete documentation of final thesis	11 days	Mon 16/11/15	Mon 30/11/15