Pairwise Testing Tools Based On Hill Climbing Algorithm (PTCA)

By

LIM SENG KEE

Thesis submitted in fulfilment of the requirements
For the degree of
BCS (Software Engineering with Honours)

Dec 2014
ABSTRACT

Failure in software can cause serious damage and because of this it is essential to test the software thoroughly. Software testing is the activity to find the defects in the components or systems. The main problem in software testing is to test the system exhaustively as it is impossible to do so. To reduce the test cases generated during testing process, combinatorial algorithms which consisting Hill Climbing algorithm and T Way Combination algorithm as described in twayGenerator (Kamal Zuhairi Zamli, 2007) have been studied and reviewed. The actual implementation of the algorithm which is in java programming language, the program is implemented on Net Bean 7.0.1. Assumptions have been made for the program. Several experiments were conducted included demonstrating the correctness of the algorithms, demonstrate PTCA successful in reducing test suite size, and benchmarking PTCA against existing strategies in order to prove PTCA achieved its objectives. Based on the experimental results, PTCA has successful in reducing the test size.
ABSTRAK

# TABLE OF CONTENTS

**SUPERVISOR'S DECLARATION** ............................................................................................................................................................ I

**STUDENT'S DECLARATION** ............................................................................................................................................................... II

**DEDICATION** ...................................................................................................................................................................................... III

**ACKNOWLEDGEMENT** ........................................................................................................................................................................ IV

**ABSTRACT** ......................................................................................................................................................................................... V

**ABSTRAK** ....................................................................................................................................................................................... VI

**TABLE OF CONTENTS** ........................................................................................................................................................................ VII

**LIST OF FIGURES** ........................................................................................................................................................................... IX

**LIST OF TABLES** ............................................................................................................................................................................... XI

## Chapter 1 INTRODUCTION .................................................................................................................................................................... 1

1.1 Software Testing ............................................................................................................................................................................ 1

1.2 Problem Statement ....................................................................................................................................................................... 2

1.3 Objective ................................................................................................................................................................................... 3

1.4 Research Scope ........................................................................................................................................................................ 3

1.5 Thesis Organization ............................................................................................................................................................... 3

1.6 Summary ................................................................................................................................................................................... 4

## Chapter 2 LITERATURE REVIEW ...................................................................................................................................................... 5

2.1 Theoretical Background ............................................................................................................................................................ 5

2.1.1 General Exhausting Testing Approach .............................................................................................................................. 5

2.1.2 Equivalence Partitioning Approach .................................................................................................................................. 7

2.2 Survey of Existing Pairwise Strategies .................................................................................................................................... 8

2.2.1 G2Way .................................................................................................................................................................................. 8

2.2.2 Genetic Algorithm (GA) .................................................................................................................................................... 10

2.3 Summary .................................................................................................................................................................................. 11

## Chapter 3 METHODOLOGY ............................................................................................................................................................... 12

3.1 Methodology ............................................................................................................................................................................... 12

3.2 Hardware & Software .............................................................................................................................................................. 14

3.3 Summary ................................................................................................................................................................................... 14

## Chapter 4 DESIGN ............................................................................................................................................................................. 15

3.1 Design Consideration .............................................................................................................................................................. 15

3.2 Development of PTCA Strategy ............................................................................................................................................. 15
<table>
<thead>
<tr>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.3</td>
</tr>
<tr>
<td>3.4</td>
</tr>
<tr>
<td>3.5</td>
</tr>
<tr>
<td>3.6</td>
</tr>
<tr>
<td>3.7</td>
</tr>
<tr>
<td>Chapter 5 IMPLEMENTATION .............................................................. 18</td>
</tr>
<tr>
<td>5.1</td>
</tr>
<tr>
<td>5.2</td>
</tr>
<tr>
<td>5.3</td>
</tr>
<tr>
<td>Chapter 6 RESULT AND DISCUSSION ....................................................... 25</td>
</tr>
<tr>
<td>6.1</td>
</tr>
<tr>
<td>6.2</td>
</tr>
<tr>
<td>6.3</td>
</tr>
<tr>
<td>6.4</td>
</tr>
<tr>
<td>Chapter 7 CONCLUSION ........................................................................ 40</td>
</tr>
<tr>
<td>7.1</td>
</tr>
<tr>
<td>7.2</td>
</tr>
</tbody>
</table>

REFERENCES ......................................................................................... 42

APPENDICES ......................................................................................... 44
LIST OF FIGURES

Figure 2.1: Algorithm for pair generation for G2Way (Klaib et al., 2008) ..............8
Figure 2.2: Example of index search (Klaib et al., 2008) .....................................8
Figure 2.3: Example of row index (Klaib et al., 2008) ..........................................9
Figure 2.4: Outline of the proposed GA (Shiba et al., 1997) ............................10
Figure 3.1: Overview flow of methodology ........................................................12
Figure 4.1: Overall design of PTCA ..................................................................15
Figure 4.2: Algorithm of hill climbing algorithm .................................................16
Figure 4.3: Algorithm of pairwise strategy ............................................................16
Figure 4.4: Algorithm of the system ....................................................................17
Figure 5.1: 4 inputs with 3 valued ......................................................................18
Figure 5.2: The possible combination of the input (1) .......................................19
Figure 5.3: The possible combination of the input (2) .......................................20
Figure 5.4: The possible combination of the input (3) .......................................21
Figure 5.5: The list when the input converting into pairwise (1) .........................22
Figure 5.6: The list when the input converting into pairwise (2) .........................23
Figure 5.7: The result of the test .......................................................................24
Figure 5.8: The common result .........................................................................24
Figure 6.1: result screen on PTCA ....................................................................26
Figure 6.2: result screen in PTCA ....................................................................29
Figure 6.3: Result of S1 ....................................................................................34
Figure 6.4: Result of S2 ....................................................................................35
Figure 6.5: Result of S4 ....................................................................................36
Figure 6.6: Result of S8 ....................................................................................37
Figure 7.1: Gantt Chart .....................................................................................44
Figure 7.2: List of pairwise generated (13 parameter with 3 value) ......................45
Figure 7.3: continue ..........................................................................................46
Figure 7.4: continue ..........................................................................................47
Figure 7.5: continue ..........................................................................................48
Figure 7.6: continue ..........................................................................................49
Figure 7.7: continue ..........................................................................................50
Figure 7.8: continue ..........................................................................................51
Figure 7.9: list of possible combination (13 parameter with 3 value) ...............52
Figure 7.10: continue ........................................................................................................53
Figure 7.11: the result (13 parameter with 3 value) ..................................................54
Figure 7.12: failure print screen during extreme huge data set .. 54
Figure 7.13: Gantt chart ............................................................................................55
**LIST OF TABLES**

Table 2.1: Example of exhaustive approach .......................................................... 5  
Table 2.2: Possible input of example ................................................................. 5  
Table 2.3: Number of test cases generated by exhaustive approach ............... 6  
Table 2.4: Test cases implement for Equivalence partitioning approach ........ 7  
Table 3.1: List of software used ......................................................................... 14  
Table 3.2: List of Hardware used ...................................................................... 14  
Table 6.1: 4 input with 3 valued ........................................................................ 25  
Table 6.2: Symbol of each input ....................................................................... 26  
Table 6.3: Result .................................................................................................. 27  
Table 6.4: List of total interaction pairs ............................................................. 27  
Table 6.5: 3 parameter with each value of 2 ..................................................... 29  
Table 6.6: Result ............................................................................................... 29  
Table 6.7: List of total interaction pairs ............................................................ 30  
Table 6.8: Input specification for test A, B, C, D, and F ................................... 31  
Table 6.9: Results of test suite size generated with number of runs ............... 31  
Table 6.10: Differences between exhaustive testing test suite size and PTT test suite size ........................................................................................................... 32  
Table 6.11: Input specification for 8 tests .......................................................... 32  
Table 6.12: Result of the comparison data in term of the test size (Klaib et al., 2008) ................................................................. 33  
Table 6.13: Results of benchmarking ................................................................ 38
Chapter 1 INTRODUCTION

1.1 Software Testing

Software testing is a process of investigate, detect, and found error or bug exist in a software is a process of validation and verification of system (Morgan P. Samaroo, A. Thompson, G. William, 2010). Software testing work in systematically to explore every possible components, systems, or flow for the purpose to found or detect the error exist in the system (Morgan P. Samaroo, A. Thompson, G. William, 2010).

Best practice in software is a much to develop a quality system, because best practice of software testing able to minimal the failure of system and it also help to maintain system marketing value. (Graham, D., Black, R., Van, V.E and Evans, 2006) Although testing make quality, but software testing is only can found the existing error but not to produce error –free system, due to this a error-free system is almost impossible because exhausting testing, test for every single components in system is impossible. Exhausting testing is impossible is impossible because it may cost a huge among of money and it also take long period to complete.

Software testing is necessary in a system development although software testing may cost a high budget, but a good quality system is needed to reduce system failure, because failure of system may cost higher profit loss than expected, is may cause damage of building, or human life. (Bryce, R.C and Colbourn, C.J., 2006) To avoid the unexpected effect from bad quality system failure, developer is trend to spend for testing process.

Due to the highly cost of testing process, both developer and stake holder are wish to have a most efficient software testing processing, which will cost minimal and found maximum defect.

In history, there are a lot of incidents due to the operation system failure. For example, European Space Agency Ariane 5 exploded incident in June 2006,
(LIONS, 1996) was causes by the software error on floating point overflow this incident was cost up to $500 million. Another incident was happen in November 2005, the United Kingdom top 10 most wanted criminals in the website was forced to be offline due to exceeding user access. All of the above failures are due the insufficient of software testing; these incidents demonstrate the important of a good software testing.

For the above reason, a new methodology of test case generator has been implemented, t-way testing method. T-way testing is a method to reduce the number of general the test cases by compromising the interaction strength. (Zamli, K. Z., Othman, R. R. and Zabil, M. H. M., 2011)

1.2 Problem Statement

Nowadays, structure of software became complex, huge data included, and multiple functions. The trend of software development is challenging the software tester on software testing process. Increase of size, complex, and function definitely will give challenge to being best practice of software testing. The increase of parameter and data will increase the number of test cases, and indirectly it also increase in cost, as mostly systems are money oriented so this trend should be avoid.

In most of the system, there are multiple data in a function and these functions are combining together to form a work system. In this case, testers are requiring writing test cases follow the possible combination of data exists in the system.

This combination of data might be huge and it may reduce by certain method. One of the exist method to reduce the number of test cases produce was T-way strategic, a strategic that delete redundancy combination data by it specific methodology.
1.3 Objective

This research thesis is aimed to develop an alternative flow test data generation for combination testing. Based on this aim, there are three objectives:

1. To develop a prototype to implement the Hill Climbing algorithm for generating test input data.
2. To investigate the correctness of the Hill Climbing algorithm implementation.
3. To evaluate the performance in terms of test size against other existing pairwise testing strategies.

1.4 Research Scope

This research thesis will cover the studies of Hill Climbing Algorithm and pairwise testing strategies. The main focus will be on the strategies to reduce test suite size using pairwise testing technique and hill climbing algorithm with achieving full coverage. NetBeans IDE 7.0.1 was used to develop this prototype.

1.5 Thesis Organization

This thesis consists of 7 chapters, every chapter covers specific issues of the research.

Chapter 1 is Introduction; in chapter 1 will cover the general view about this research. For example, introduction, problem statement, objective, and research scope, are all included in the chapter 1.

Chapter 2 is the Literature Review; in chapter 2 all related issues with the research will be discussed here. In addition, chapter 2 will review on the existing method.

Chapter 3 is Methodology; in chapter 3 the flow of the research will be discussed and Gantt Chart of estimation on the date will be attached to show a tangible planning of work flow.

Chapter 4 is about design, in design stage the detail design of the thesis prototype will plan and show. The algorithm of how the prototype flow and function will be fully described in this chapter.
Chapter 5 is Implementation, in this chapter development of prototype will be start and done follow the design on chapter 4. The prototype developed should be functional to enable the next stage of research.

Chapter 6 is Result & Discussion, in this stage a numbers of data will be tested with the developed prototype, and the result will be record and discuss, compare, and verify with the expected result. In begin of this chapter expected result should be define.

Chapter 7, Conclusion will be last and end of research, a final result of the research should be stated and a conclusion with detail explanation of the research should be written.

1.6 Summary

This chapter has discussed the introduction of research on Pairwise Testing Tools Based on Hill Climbing Algorithm. Problem statement, objective, research scope, thesis organization was included in this chapter.
Chapter 2 LITERATURE REVIEW

In Chapter 1, the importance of software testing has been discussed. Building on the material in Chapter 1, this chapter presents the relevant literature review survey. In particular, details of the test case design techniques, existing pairwise testing strategy, covering arrays, and Java script are elaborated in order to justify the current work.

2.1 Theoretical Background

In order to highlight the current work into perspective, this section describes the existing techniques to design the test cases. The techniques that are included in this section are exhausting approach, equivalence partitioning, boundary value analysis, decision table and pairwise testing.

2.1.1 General Exhausting Testing Approach

Exhausting testing was the general method on generating test cases, exhausting approach basically is an approach that will generate all possible test cases for a system. Due to this characteristic, exhausting is always impossible for a huge system, because it is impossible to test all combination. It is costly and time consuming. As example in a restaurant, there is a set menu that included 4 types of foods, each food has several favours.

Table 2.1: Example of exhaustive approach.

<table>
<thead>
<tr>
<th>Input</th>
<th>Symbolic Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Burger</td>
<td>B</td>
</tr>
<tr>
<td>Drink</td>
<td>D</td>
</tr>
<tr>
<td>Ice-Cream</td>
<td>I</td>
</tr>
<tr>
<td>French Fries</td>
<td>F</td>
</tr>
</tbody>
</table>

Table 2.2: Possible input of example.

<table>
<thead>
<tr>
<th>Symbolic Representation</th>
<th>Possible Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>B1, &amp; B2</td>
</tr>
<tr>
<td>D</td>
<td>D1, &amp; D2</td>
</tr>
<tr>
<td>I</td>
<td>I1, I2, &amp; I3</td>
</tr>
<tr>
<td>F</td>
<td>F1, F2, &amp; F3</td>
</tr>
</tbody>
</table>
Table 2.1 and Table 2.2 is the list of the possible input for this system. And the numbers test cases generated should be $2\times2\times3\times3 = 36$. Table 2.3 has shown all the 36 possible test cases, but there is too much test cases included in this case.

*Table 2.3: Number of test cases generated by exhaustive approach.*

<table>
<thead>
<tr>
<th>Burger</th>
<th>Drink</th>
<th>Ice-Cream</th>
<th>French fries</th>
</tr>
</thead>
<tbody>
<tr>
<td>B1</td>
<td>D1</td>
<td>I1</td>
<td>F1</td>
</tr>
<tr>
<td>B1</td>
<td>D2</td>
<td>I1</td>
<td>F1</td>
</tr>
<tr>
<td>B1</td>
<td>D1</td>
<td>I2</td>
<td>F1</td>
</tr>
<tr>
<td>B1</td>
<td>D2</td>
<td>I2</td>
<td>F1</td>
</tr>
<tr>
<td>B1</td>
<td>D1</td>
<td>I3</td>
<td>F1</td>
</tr>
<tr>
<td>B1</td>
<td>D2</td>
<td>I3</td>
<td>F1</td>
</tr>
<tr>
<td>B1</td>
<td>D1</td>
<td>I1</td>
<td>F2</td>
</tr>
<tr>
<td>B1</td>
<td>D2</td>
<td>I1</td>
<td>F2</td>
</tr>
<tr>
<td>B1</td>
<td>D2</td>
<td>I2</td>
<td>F2</td>
</tr>
<tr>
<td>B1</td>
<td>D1</td>
<td>I2</td>
<td>F2</td>
</tr>
<tr>
<td>B1</td>
<td>D2</td>
<td>I3</td>
<td>F2</td>
</tr>
<tr>
<td>B1</td>
<td>D1</td>
<td>I3</td>
<td>F2</td>
</tr>
<tr>
<td>B1</td>
<td>D2</td>
<td>I1</td>
<td>F3</td>
</tr>
<tr>
<td>B1</td>
<td>D2</td>
<td>I1</td>
<td>F3</td>
</tr>
<tr>
<td>B1</td>
<td>D1</td>
<td>I2</td>
<td>F3</td>
</tr>
<tr>
<td>B1</td>
<td>D2</td>
<td>I2</td>
<td>F3</td>
</tr>
<tr>
<td>B1</td>
<td>D1</td>
<td>I3</td>
<td>F3</td>
</tr>
<tr>
<td>B1</td>
<td>D2</td>
<td>I3</td>
<td>F3</td>
</tr>
<tr>
<td>B2</td>
<td>D1</td>
<td>I1</td>
<td>F1</td>
</tr>
<tr>
<td>B2</td>
<td>D2</td>
<td>I1</td>
<td>F1</td>
</tr>
<tr>
<td>B2</td>
<td>D1</td>
<td>I2</td>
<td>F1</td>
</tr>
<tr>
<td>B2</td>
<td>D2</td>
<td>I2</td>
<td>F1</td>
</tr>
<tr>
<td>B2</td>
<td>D1</td>
<td>I3</td>
<td>F1</td>
</tr>
<tr>
<td>B2</td>
<td>D2</td>
<td>I3</td>
<td>F1</td>
</tr>
<tr>
<td>B2</td>
<td>D1</td>
<td>I1</td>
<td>F2</td>
</tr>
<tr>
<td>B2</td>
<td>D2</td>
<td>I1</td>
<td>F2</td>
</tr>
<tr>
<td>B2</td>
<td>D2</td>
<td>I2</td>
<td>F2</td>
</tr>
<tr>
<td>B2</td>
<td>D1</td>
<td>I2</td>
<td>F2</td>
</tr>
<tr>
<td>B2</td>
<td>D2</td>
<td>I3</td>
<td>F2</td>
</tr>
<tr>
<td>B2</td>
<td>D1</td>
<td>I3</td>
<td>F2</td>
</tr>
<tr>
<td>B2</td>
<td>D1</td>
<td>I1</td>
<td>F3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Burger</th>
<th>Drink</th>
<th>Ice-Cream</th>
<th>French fries</th>
</tr>
</thead>
<tbody>
<tr>
<td>B2</td>
<td>D1</td>
<td>I1</td>
<td>F3</td>
</tr>
</tbody>
</table>
2.1.2 Equivalence Partitioning Approach

Equivalence partitioning testing approach is a black box test cases approach, which means only concern on the input and output. Equivalence partitioning will classified into group according to specific characteristic. For each group, only a set of data will be tested due to the similarity of the data. For example:

\[-5 < a < 40\]

\[-12 < b < 35\]

\[0 < c < 20\]

The test cases used will be:

*Table 2.4: Test cases implement for Equivalence partitioning approach.*

<table>
<thead>
<tr>
<th>Test</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-4</td>
<td>-13</td>
<td>-1</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>10</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>45</td>
<td>50</td>
<td>22</td>
</tr>
</tbody>
</table>

In this approach, the coverage of the flow is very small, so it is not a good testing approach, especially for the cases that contain alternative flow. But this approach has its benefits too, for example it consume least time and cost.
2.2 Survey of Existing Pairwise Strategies

After completed on the theoretical background, the followed section was surveying on existing pairwise strategies.

2.2.1 G2Way

G2Way is one of the tools that apply pairwise testing strategy, G2Way consist of pair generation algorithm and backtracking algorithm (Klaib et al., 2008). Figure 2.1 show the algorithm for pair generation for G2Way strategy.

```
Algorithm PairGeneration()
1: begin
2: initialize Sp=\{\} where Sp represents the pair set
3: let \( n_{C} = \{n_{1}, \ldots, n_{m}\} \) where \( n_{C} \) represents the values defined for each parameter, \( m = \text{maximum no of parameters} \)
4: let \( p = \{p_{0}, \ldots, p_{m}\} \), where \( p \) represents the sorted set of sets of values defined for each parameter
5: for index=0 to \( 2^{m} - 1 \)
6: begin
7: let \( b = \text{binary number} \)
8: if (the no of '1's in b= 2)
9: begin
10: calculate number of possible combinations (PC) between the partial sets of values
11: for the shared parameters
12: begin
13: multiply \( \{n_{x}, n_{y}\} \) values from \( n_{C} \)
14: set the bits group (equal to PC) in the index row to 1
15: end
16: end
17: end
18: return \( S_{p} \)
19: end
```

Figure 2.1: Algorithm for pair generation for G2Way (Klaib et al., 2008)

Based on the figure, the loop edge for 2-way interaction will be finding first. Then, the index search will be performed. Considers there are 3 parameters P0, P1, P2 where each parameters have value 2, 3, 1 respectively, the loop search will be 7 where \( 2^{3} - 1 \). The index’s number will be converting to binary format as in Figure 2.2.

<table>
<thead>
<tr>
<th>Index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary</td>
<td>000</td>
<td>001</td>
<td>010</td>
<td>011</td>
<td>100</td>
<td>101</td>
<td>110</td>
<td>111</td>
</tr>
</tbody>
</table>

Figure 2.2: Example of index search (Klaib et al., 2008)

The index with two binary one’s will be put in the index set. Based on the example in the figure above, index 3, 5, and 6 have two binary ones. Then, each row of
combination of possible pairwise value is generated. There are 3 rows of possible pairwise combination which are P0 and P1, P0 and P2, and P2 and P3.

<table>
<thead>
<tr>
<th>Row Index</th>
<th>Index</th>
<th>b5</th>
<th>b4</th>
<th>b3</th>
<th>b2</th>
<th>b1</th>
<th>b0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

*Figure 2.3: Example of row index (Klaib et al., 2008)*

Row index 0 is the combination of P0 and P1. The value for P0 is 2 and the value for P1 is 3. Thus the row index store 6 pairs of combination. For row index 1 and 2, 2 pairs stored respectively.

To complete a test suite backtracking algorithm crosses the set of pairwise in iterative ways to combine pairs with usual values of parameter. This algorithm will start to the first define values if pairs cannot be combined existed. The values in pairwise set deleted when the pairs are covered, all pairs are covered only if the pairwise set is empty.
2.2.2 Genetic Algorithm (GA)

Shiba et al. (2004) proposed two new algorithms based on two artificial life techniques to generate test case for combinatorial testing. One of the proposed algorithms is AETG algorithm modification with Genetic Algorithm (GA).

```
Input: A test set.
Output: A test case.
Begin
    Create the initial population $I'$ of candidates.
    Evaluate $I'$.
    While (stopping condition is not met) {
        Select Elite consisting of $\sigma$ best individuals from $I'$.
        Apply Selection to individuals in $I'$ to create $P_{mating}$, consisting of $(m - \sigma)$ individuals.
        Crossover $P_{mating}$.
        Mutate $P_{mating}$.
        Copy the all individuals of $P_{mating}$ to $P'$, replacing the worst $(m - \sigma)$ individuals in $P$.
        Evaluate $I'$.
        If (stagnation condition is met) Mutate $I'$ massively.
    }
    Return the best test case found.
End.
```

Figure 2.4: Outline of the proposed GA (Shiba et al., 1997)

GA impersonates the growth of single celled organism. Generally there are four important aspects in GA that are chromosome encoding and fitness function, selection, crossover and mutation, and GA treated a test case as a chromosome, and the fitness function is used to approximate the goodness of the candidate solution. Fitness function $F(S)$ for a test case $S$ is defined as the number of $t$-way combinations that covered by $S$ but not covered by the given test set. Based on the algorithm shows in the figure, the initial population of test cases’ candidate, $P$ is created randomly.

The population is then evaluated. The best test case from the population is selected using elite strategy. To the remaining test case, Selection is applied to create
population mating. After that, the test case in mating pool is being crossover and the value of a position is replaced with another value by mutation randomly. After crossover and mutation, the test cases in mating pool are copy to the P to replace the worst test case. Then, the P is evaluated again. P is mutated vastly if the stagnation condition is met. In this case, the stagnation condition is the generations' number that passes ever since the last improvement in solution. These actions are performed continuously until the stopping condition met.

2.3 Summary

This chapter has review on the existing testing theoretical background and general idea of pairwise testing, and existing pairwise testing strategy.
Chapter 3 METHODOLOGY

In chapter 2, the review on the existing testing strategies has been discussed. In this chapter, the methodology of PTCA has been discussed in detail.

3.1 Methodology

Research methodology is an explanation of the process included to complete the research project. There are four stages included in this research project, for example, Literature Review, Analysis and Design, Development, and Analysis the result. As shown in figure 3.1.

![Figure 3.1: Overview flow of methodology](image)

The first stage, Literature Review is a stage where will study of the existing problem on the past testing process faced. The concern on this stage is about the t-way testing strategies, and how this strategies help in reduce the number of test cases.
Secondly, we will proceed into Analysis and Design process. In here we are going to analyze the possible strategies to improve the testing process. In addition here also will include the design of the new possible strategies to reduce the number of test cases.

Next process is the development stage; this stage is mainly focused on to implement the possible strategies. Furthermore, in this stage test cases generator will be develop base on the new strategies.

The final stage was the Analysis Result process, in here collecting of the result producing by the tool (test case generator) and the analysis of these result will be proceed. Moreover, the result of this research will be answer in this stage.
3.2 Hardware & Software

Table 3.1: List of software used

<table>
<thead>
<tr>
<th>Software Used</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Net Bean</td>
<td>Use in development the prototype in Java.</td>
</tr>
<tr>
<td>Microsoft Office</td>
<td>Use in documentation, e.g writing report,</td>
</tr>
<tr>
<td></td>
<td>drawing chart and e.t.c</td>
</tr>
</tbody>
</table>

Table 3.2: List of Hardware used

<table>
<thead>
<tr>
<th>Hardware Used</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Laptop</td>
<td>Use in development the prototype in Java.</td>
</tr>
<tr>
<td></td>
<td>Use in writing report.</td>
</tr>
<tr>
<td>Stationary</td>
<td>Use to record minute of meeting.</td>
</tr>
<tr>
<td></td>
<td>Use on drafting.</td>
</tr>
</tbody>
</table>

Table 3.1 & table 3.2 has listed the software and hardware used for PTCA. PTCA was developed using java programing language with NetBean as platform. During the research, MS office was used as report writing, result recording.

For hardware, an Intel i5-2410M, 2.30GHz with 8GB ram laptop was used to develop the prototype and report writing.

3.3 Summary

This chapter discussed on the flow of the PTCA research, besides hardware and software used has been included too.
Chapter 4 DESIGN

In chapter 3, detail of the tools and research flow has been discussed. In this chapter, detail design of PTCA has been discussed.

3.1 Design Consideration

The system designed is selected the loop from first test case in the test cases, and get the coverage of each test case, then save the most coverage test case as the best test case.

3.2 Development of PTCA Strategy

PTCA is tools that function based on pairwise testing strategy and hill climbing algorithm. Figure 4.1 show the overall design of PTCA.

![PTCA Design Diagram]

Figure 4.1: Overall design of PTCA

PTCA strategy design used pairwise testing strategy to reduce the test case generated, based on the hill climbing algorithm to as process method, to achieve full coverage of the combination.
3.3 Hill Climbing Algorithm

```
1. for (int i=0; i<sample.length; i++)
2.     for (int r=i; r<sample.length; r++)
3.         compare sample[i] with sample[r]
4.     if sample[i] < sample[r]
5.         temp = sample[i]
6.         sample[i] = sample[r]
7.         sample[r] = temp
8. end
```

Figure 4.2: Algorithm of hill climbing algorithm

Hill climbing algorithm is an incremental concept algorithm which would start from the basic to the peak. On PTCA, hill climbing algorithm was used on merging process. Which PTCA will retrieve the 1st combination from the combination list to merge with the data in the pair list.

3.4 Weight Counting Method

Weight counting is a identified in PTCA, for each loop of merging process the weight will recorded and compare with the previous weight. The heaviest weight combination consider as the best.

3.5 Pairwise Testing Strategy

```
1. Begin
2. let pair as ArrayList[]
3. let pair=[-1,-1,-1-1,...]
4. let int time =0
5. for (int s1=0; s1< pair.length; s1++){
6.     for (int s2=1; s2< pair.length; s2++){
7.         for (int value1=0; value1< parameter_1; value1++){
8.             for (int value2=0; value2< parameter_2; value2++){
9.                 replace pair.[time].[s1]= value_1
10.                replace pair.[time].[s2]= value_2
11.            time++
12.         }}
13. }}
End.
```

Figure 4.3: Algorithm of pairwise strategy