CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

Nowadays, the era of advanced technology has brought modern software systems into our daily life to satisfy the various needs of users. To ensure a well function of software systems without defects which can leads to failure and malfunction, software testing plays a dominant role on it. It has been known for some time that, in a typical programming project, approximately 50% of elapsed time and over 50% of total cost are expended in testing part (Dalley, 1991).

According to definition of Institute of Electrical and Electronics Engineers (IEEE), Software Testing is a process to analyze a software system to determine the differences between the requirements and the existing condition of the software like defects or bugs and evaluate the defects found in the software system. In other words, the main objective of software testing is to perform the verification of software system so that it is well-developed according to the requirements and free from failure caused by defects.

Combinatorial testing (CT) is one of the testing strategy which selecting test case by combining the values of different input parameters using some combinatorial algorithm strategy (Mats Grindal, 2004). However, it is often impractical to execute exhaustively by testing all the combinations due to complexity, time-consuming, resource constraints and require a huge amount of cost as well (Shiwei Gao, 2014). The size of a test list required to test all of the possible combinations can be impossible in even a moderate size of project.
(Myra B. Cohen, 2003). Furthermore, to generate a combinatorial test list which consists of more than 6-ways combination is very expensive also (Kiran Shakya, 2012).

Therefore, it is crucial to propose another strategy to reduce the number of test cases in order to reduce the cost and time needed in software testing. With this, t-way combinatorial testing which generate test case covered all t combinations of values of every parameters of a System Under Test (SUT) is encouraged (Shiwei Gao, 2014). This testing strategy generated smaller number of test list based on t-way combinations of input values of t parameters where t also represents the strength of interaction. Each t-way combination of valid values of these t parameters is covered by at least one test case (R. K. Yu Lei, D.Richard Kuhn, Vadim Okun, James Lawrence, n.d.).

There are many existing t-way strategies which includes pure computational-based like Test Vector Generator (TVG) (P.J. Schroeder, 2003) , ParaOrder (Z.Y. Wang, 2008) and Artificial Intelligence (AI) – based like Genetic Algorithm (McCaffrey, 2010; R. Bryce, 2007; Sthamer, 1995; Toshiaki Shiba, 2004; W. Afzal, 2009) and Harmony Search Algorithm (Abdul Rahman A. Alsewari, 2012). With this, this research proposed t-way Testing : A Test Case Generator based on another algorithm under Artificial Intelligence (AI) based called Melody Search algorithm which will be further discussed in details in Chapter 4.
1.2 PROBLEM STATEMENT

During testing, we need to optimize or reduce the test cases so that we do not repeat certain interaction of values. This problem is considered as Combinatorial Optimization (CP) Problem. Combinatorial optimization problem involves the process of finding optimal solutions or minimum number of test suite.

For example, TSY food ordering system consists of 5 parameters with 2 three-valued parameters and 3 two-valued parameters as shown in Table 1. To test all the possible combinations exhaustively, the number of test case will be $3^2 \times 2^3 (v^p)$ where $v$ refers to the number of values and $P$ refers to the number of parameters, so $3^2 \times 2^3 = 72$ test case. In other words, if the system continues to expand to become 10 parameters with 7 three-valued parameters and 3 two-valued parameters, the number of test case will become $3^7 \times 2^3 = 17496$ test cases. It will become impossible as it require a huge amount of cost and time to test.

<table>
<thead>
<tr>
<th>Food</th>
<th>Flavor</th>
<th>Add Ingredients</th>
<th>Delivery</th>
<th>Payment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mee</td>
<td>Spicy</td>
<td>Egg</td>
<td>Eat in</td>
<td>Cash</td>
</tr>
<tr>
<td>Fried Rice</td>
<td>Not Spicy</td>
<td>Chicken</td>
<td>Take Away</td>
<td>Credit Card</td>
</tr>
<tr>
<td>Meehun</td>
<td></td>
<td>None</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>