T-Way Test Suite Generator Based on Gravitational Search Algorithm

by

Khin Maung Htay
(1930213134)

A thesis submitted in fulfillment of the requirements for the degree of Master of Science (Computer Engineering)

Faculty of Electronic Engineering Technology
UNIVERSITI MALAYSIA PERLIS

2021

ACKNOWLEDGEMENT

First and foremost, in the name of Allah (S.W.T), the Most Gracious, the Most Merciful, I am grateful to Allah (S.W.T) for keeping me healthy and for awarding me with the capability to complete my MSc research project successfully. Secondly, my heartfelt thanks go to my supervisor, Assoc. Prof. Dr. Rozmie Razif Othman, who is willing to give me valuable advice, guidance, and explanations about the project. Without his help, I would not have been able to complete it within this short period of time. I would also like to thank my co-supervisor, Assoc. Prof. Ts. Dr. Amiza Amir, for her constant support and encouragement from the beginning until the end of my study period. Not to forget, I am thankful to Ts. Dr. Hasneeza Liza Zakaria for sharing with me valuable resources regarding statistical analysis tests.

Thirdly, I would like to express my gratitude towards my family, especially my beloved parents, uncle, and aunty, for supporting and encouraging me not to give up and to keep trying and working despite the difficulties that I encountered. I also would like to thank Universiti Malaysia Perlis (UniMAP) and the Faculty of Electronic Engineering Technology (FTKEN) for giving me this excellent opportunity to practice the knowledge I had gained throughout my years of study.

Last but not least, much appreciation goes to those who have helped me out with their abilities directly or indirectly in completing this project.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABC	Artificial Bee Colony
ABC-TG	Artificial Bee Colony Test Generator
AI	Artificial Intelligence
AETG	Automatic Efficient Test Generator
BVA	Boundary Value Analysis
CPU	Central Processing Unit
CFO	Central Force Optimization
CS	Cuckoo Search
CSS	Cuckoo Search Strategy
CA	Covering Array
DFA	Dragonfly Optimization Algorithm
DF	Degree of Freedom
FPA	Flower Pollination Algorithm
FB	Filtered Blood
FTS	Final Test Suite
GSA	Gravitational Search Algorithm
GSTSG	Gravitational Search Test Suite Generator
HS	Harmony Search
IPOG	In-Parameter-Order-General
IPOG-D	In-Parameter-Order-General Double
IPOF	Heuristic Based IPOG
ITTDG	Integrated T-Way Test Data Generation
IJA	Improved Jaya Algorithm
IDE	Integrated Development Environment
IPMBOS	Improved Pairwise Migrating Birds Optimization Strategy
IMTS	Improved Migrating Birds Optimization Testing Strategy
ISTQB	International Software Testing Qualifications Board
JA	Jaya Algorithm
JDK	Java Development Kit
JVM	Java Virtual Machine
KA	Kidney Algorithm
LHS-JA	Latin Hypercube Sampling Jaya Algorithm
MAETG	Modified Automatic Efficient Test Generator

MBO	Migrating Birds Optimization
NA	Not Available
NFL	No Free Lunch
NS	Not Supported
OA	Orthogonal Array
OTAT	One Test at a Time
OPAT	One Parameter at a Time
PSO	Particle Swarm Optimization
PSTG	Particle Swarm Test Generator
PKS	Pairwise Kidney Strategy
PICT	Pairwise Independent Combinatorial Testing
QLSCA	Q-Learning Sine Cosine Algorithm
RAM	Random Access Memory
SUT	System Under Test
SA	Simulated Annealing
STLC	Software Testing Lifecycle
SOTA	State of the Art
SCA	Sine Cosine Algorithm
TL	Tuple List
TCONFIG	Test Configuration
WOA	Whale Optimization Algorithm
W	Waste Group

LIST OF SYMBOLS

λ	Index of Orthogonal Array
H_1	Alternative Hypothesis
H_0	Null Hypothesis
χ^2	Chi-Square Value
α	Confidence Interval
p	Conditional Probability or Critical Value
t	Interaction Strength

Penjana Uji Suite T-Way Berdasarkan Algoritma Pencarian Graviti

ABSTRAK

Disebabkan oleh keperluan pengguna yang berbeza, perisian kontemporari telah menjadi kaya dengan ciri dari segi fungsi input (iaitu, parameter) dan pilihan (iaitu, nilai). Ujian menyeluruh pada sistem perisian yang canggih adalah tidak praktikal setakat masa dan kos ujian. Pelbagai teknik ujian perisian seperti pembahagian kelas kesetaraan, analisis nilai sempadan dan jadual keputusan, telah dicadangkan dalam literatur dengan tujuan tunggal untuk mengesan kesilapan faktor tunggal. Tidak seperti kerja-kerja terdahulu, ujian kombinatorial t-way (di mana t menunjukkan kekuatan interaksi/gabungan) menyokong pengesanan ralat yang disebabkan oleh dua atau lebih interaksi parameter sistem input (iaitu, ralat berbilang faktor) sambil meminimumkan saiz suite ujian dengan cekap. berbanding dengan senarai ujian yang lengkap. Sejak beberapa tahun kebelakangan ini, algoritma pengoptimuman metaheuristik nampaknya menjadi pilihan yang paling biasa untuk penyelidik memandangkan keberkesanannya terbukti menawarkan hasil yang optimum/hampir optimum. Walau bagaimanapun, memandangkan penjanaan suite ujian t-way ialah Non-Deterministic Polynomial Time Hard (masalah NP-hard), tiada strategi t-way tunggal boleh menjamin keunggulan daripada yang lain untuk semua jenis konfigurasi sistem. Oleh itu, tesis ini membentangkan strategi t-hala kekuatan seragam (kekuatan interaksi tetap) baharu berdasarkan Algoritma Carian Gravitational (GSA), yang dipanggil Gravitational Search Test Suite Generator (GSTSG). Sumbangan utama tesis penyelidikan ini ialah strategi GSTSG yang dicadangkan menggunakan GSA buat kali pertama untuk penjanaan data ujian t-way, yang masih belum diterokai dalam penyelidikan ujian perisian. Keputusan penanda aras menunjukkan bahawa GSTSG memperoleh hasil yang kompetitif dalam kebanyakan konfigurasi sistem jika dibandingkan dengan strategi sedia ada yang lain dan menangani liputan interaksi yang lebih tinggi sehingga t = 10 disebabkan oleh ciri tanpa ingatan GSA. Secara keseluruhan, GSTSG menghasilkan saiz suite ujian yang optimum dalam 56.25% konfigurasi sistem daripada sejumlah 64 konfigurasi yang ditanda aras. Selain itu, keputusan yang diperolehi oleh GSTSG adalah signifikan secara statistik daripada strategi lain dalam semua perbandingan berbilang menggunakan ujian Friedman dan juga menunjukkan perbezaan dalam 9 ujian berpasangan antara 15 jumlah perbandingan berpasangan mengikut ujian Wilcoxon Signed Ranks.

T-Way Test Suite Generator Based on Gravitational Search Algorithm

ABSTRACT

Due to different users' requirements, contemporary software has become feature-rich in terms of input functions (i.e., parameters) and selections (i.e., values). Exhaustive testing on sophisticated software systems is impractical as far as testing time and cost are concerned. Various software testing techniques such as equivalence class partitioning, boundary value analysis and decision table, have been proposed in the literature with the sole purpose of detecting single-factor faults. Unlike earlier works, combinatorial t-way testing (where t indicates the interaction/combination strength) supports the detection of faults caused by two or more input system parameter interactions (i.e., multi-factor faults) while efficiently minimizing the size of the test suite as compared to exhaustive test lists. Over the past few years, metaheuristic optimization algorithms have appeared to be the most common choice for researchers since their effectiveness proves to offer optimal/near-optimal results. However, as the t-way test suite generation is a Non-Deterministic Polynomial Time Hard (NP-hard problem), no single t-way strategy can guarantee superiority to others for all types of system configurations. Hence, this thesis presents a new uniform strength (fixed interaction strength) t-way strategy based on Gravitational Search Algorithm (GSA), called Gravitational Search Test Suite Generator (GSTSG). The key contribution of this research thesis is that the proposed GSTSG strategy employs GSA for the first time for t-way test data generation, which has yet to be explored in software testing research. The benchmarking results showcase that GSTSG obtains competitive results in most system configurations when compared to other existing strategies and addresses higher interaction coverages of up to t = 10 owing to the memoryless feature of GSA. Overall, GSTSG produces optimal test suite sizes in 56.25% of system configurations out of a total of 64 benchmarked configurations. Moreover, the results obtained by GSTSG is statistically significant from other strategies in all multiple comparisons using the Friedman test and also shows differences in 9 paired tests among 15 total pairwise comparisons according to the Wilcoxon Signed Ranks test.

CHAPTER 1 : INTRODUCTION

1.1 Research Background

Software nowadays has become influential to the daily lives of people. People can no longer imagine the world without software. As an instance, we use software on a daily basis without noticing the sound of an alarm from our smartphone or digital watch, which has already included software all the way up to the point where we begin working with the computer, which is fully guided by the combination of hardware and software. For these reasons, contemporary software is developed to make it customizable in order to meet the different customers' needs (Baresi et al., 2006). However, on the other hand, it happens to be more complex concerning design, size, functionality, and involvement of the number of inputs. Here, software complexity primarily includes the interaction between one piece of code and other code pieces, which yields the likelihood of creating bugs in the system known as interaction failures (Kuhn et al., 2010).

Thus, the process of checking the quality of the developed software, which is known as software testing, plays an essential role in the lifecycle of software development (Uddin & Anand, 2019). Software testing involves the process of finding errors or bugs in the software system under test (SUT) to ensure that the particular software meets its required specifications. Successful testing is the one that can detect errors and can verify that the tested software can perform what it is supposed to be doing. Lack of testing in the software development process could lead to failures in big projects and mission-critical software that could even claim lives in some extreme cases (Leveson, 2004).

As software typically consists of multiple inputs with varying configurations, software test designs are needed to generate quality test cases called test suite. In the literature, test case design strategies such as equivalence class partitioning, boundary value analysis, decision tables, and combinatorial testing have been proposed (Bhat & Quadri, 2015) (Sawant, Bari, & Chawan, 2012) (D. R. Kuhn et al., 2015).

The equivalence class partitioning strategy is utilized to partition the test condition or the values of the input parameters into a set of equivalence classes. The test cases are chosen from those classes, with a single test case from each class, assuming that all other input values from that particular class will work equivalently. Hence, if a bug is detected by a test case in a class partition, it is more likely that all other test cases in that specific class partition detect the same bug. This strategy reduces the program's testing time as if the entire program is exhaustively tested within the same equivalence class, even if only one input value from that class is being tested. Figure 1.1 below provides a simple description of the use of the technique.

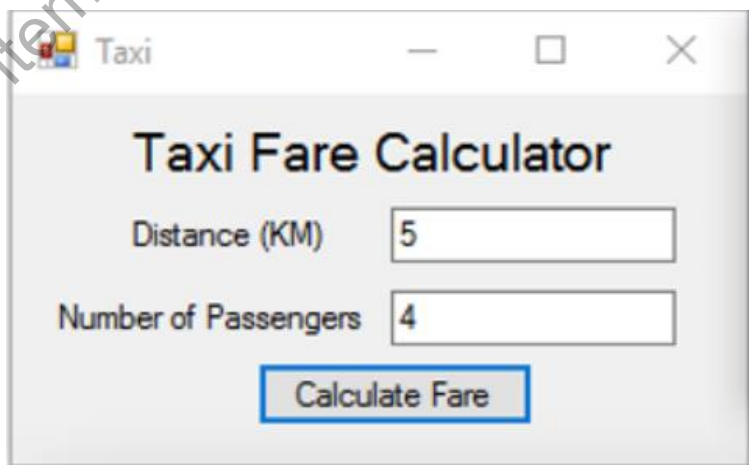

Figure 1.1 A Simple Application to Express the Use of Equivalence Partitioning

Figure 1.1 shows a simple application to calculate the taxi fare based on the distance traveled and the number of passengers included. For instance, if the distance is more than 5km, 25% of the cost and if it is more than 10km, 50% of the cost will be added to the total fare respectively. As for the number of passengers, if the number is more than 5 persons, 50% of the cost will be added to the total. Although this program can be tested by an exhaustive testing technique, the testing time will be costly. Due to this reason, the equivalence class partitioning strategy is applied here to partition the values of the distance and number of passengers into specific classes. There will be three classes for the distance in this application, the distance below 5km, the distance between 5-10km, and the distance above 10km. And there will also be two classes for the number of passengers, the number of passengers fewer than 4 people and the numbers more than 4. In order to test the total fare function, it can be chosen one value from each class partition, especially the middle value. In the case of the distance, 2, 7, and 15 and regarding the number of passengers, 2 and 5 are required to be tested.

Although the equivalence class partitioning strategy minimizes redundancy and produces good results in less time by selecting test cases from equivalence classes, bugs can also be detected at the partition boundaries of equivalence classes. Thus, the boundary value analysis (BVA) strategy, which selects the boundary values of the input ones in the class, is developed to supplement the equivalence class partitioning strategy. When determining the boundary values, the values can be included inside or outside boundaries. Applying BVA in the example application in Figure 1.1, for the distance, it can be considered selecting (-1,0,1), (4,5,6) and (9,10,11) and for the number of passengers, (-1,0,1) and (3,4,5) as test cases.

The decision table technique is used to test the different input combinations of the software system. It is also called cause and effect graphing as a set of causes or input conditions, and a set of effects or the expected outputs are captured and presented in rows and columns table form. The test cases are then generated based on the combinations of causes and effects ("Decision Table Testing: Learn with Example," n.d.) (Sawant et al., 2012).

Although the decision table testing helps better test coverage from exhaustive test cases, the table can become complicated as the number of input conditions rises. As a result, combinatorial testing (also known as t-way testing) has emerged as another type of test case design strategy that pays attention to finding bugs due to interaction failures in the system under test (SUT) (Zamli et al., 2011) (Othman & Zamli, 2011). This type of testing focuses on testing interactions of parameters rather than individual parameter testing that previous test design techniques have favored. The test suite is systematically generated based on t-way strength coverage (i.e., t value can range from 2 to the total number of involved parameters in the SUT). Each generated test case from the test suite can cover the required parameter interactions. Moreover, the test cases produced by the t-way strategy are also minimized compared to exhaustive test lists when the interaction strength is relaxed. According to the research done by (Rick Kuhn, Lei, & Kacker, 2008), 70 to almost 100 percent of faults are detected by two to six-way interactions upon testing across diverse system domains and thus, considered efficient testing strategy. Nevertheless, another study argues that testing until t = 6 is not enough for sophisticated systems as added features can introduce faults required to test with t > 6 interaction strength (Younis, Zamli, & Isa, 2012).

Different approaches of combinatorial t-way strategies have been adopted for test suite generation, such as algebraic, pure-computational, and metaheuristics. Nevertheless, preceding studies show that metaheuristic optimization algorithms inspired by the real-world natural phenomenon yield better results regarding producing the optimum test suite (M. Almufti, 2019).

Gravitational Search Algorithm (GSA) is a physics-based metaheuristic algorithm created by Iranian computer scientists to solve various kinds of optimization problems in engineering, including, but not limited to, electrical power control and management, wireless sensor network (WSN) node clustering and routing, fine-tuning the fuzzy control for servo systems, pattern recognition in image processing and training deep learning neural network architectures. To describe one use case of GSA in a study (Ezzat, Hassanien, & Ella, 2021), as an example, GSA is applied to provide the optimal value ranges for the hyperparameters of the popular 121-layer DenseNet deep learning model for diagnosing Covid-19 disease through images of chest x-rays. Even though researchers have used GSA for their successful optimization tasks, however, the algorithm has not yet been explored by the software testing community. Thus, to fill up this gap, this research work plans to develop a t-way test generation strategy based on Gravitational Search Algorithm (GSA). The upcoming chapter provides the introduction and explanation on GSA (i.e., refer to Chapter 2, Section 2.5 for details).

1.2 Problem Statements

With the increase of complexities and functionalities in modern-day integrated software systems, there are more likely to find bugs due to interactions between the inputs. Thus, exhaustive testing or complete testing is applied in the literature to test all

possible input interactions in the whole system. Even though this technique is designed to test all potential exhaustive test cases, it is, in fact, impractical and is deprived of cost efficiency as a result of having to test the uncountable number of test cases for real-life complex software systems (Nie & Leung, 2011)(Hu, Wong, Kuhn, & Kacker, 2020).

The primary challenge of software testing is to generate test cases (test suite) that cover all essential features of the SUT so that test execution can be performed in an adequate time and cost. Because of why, various test case design techniques, which consist of equivalence class partitioning, boundary value analysis, and decision tables (Bhat & Quadri, 2015)(Sawant et al., 2012)(Hübner, Huang, & Peleska, 2019), have been described in the past. Even though these techniques are helpful, they cannot effectively deal with bugs caused by interaction (Calvagna, Gargantini, & Tramontana, 2009)(Alazzawi, Rais, & Basri, 2020)

Therefore, researchers have introduced combinatorial test strategies known as t-way strategies to complement the techniques mentioned above. Researchers also use the term uniform strength testing interchangeably for t-way strategies as the mechanism uniformly covers the same interaction of all system parameters in the SUT. The critical goal of t-way testing is to effectively generate test cases that cover all the required parameter interactions, and the strategy produces lesser test cases compared to exhaustive test cases based on the choice of interaction strength. To fulfil this goal, pure-computational t-way strategies such as Jenny, Integrated T-Way Test Data Generation (ITTDG), In-Parameter-Order (IPO), Automatic Efficient Test Generator (AETG) ;to name a few, have been proposed that can support interaction strength (t combination

strength) ranging from 2 to 6 (Ahmed, Enoiu, Afzal, & Zamli, 2020)(Nasser, Zamli, Alsewari, & Ahmed, 2018).

While these strategies produce satisfactory results, studies (Alsewari & Zamli, 2012) show that artificial intelligence (AI) based t-way strategies yield better results as far as generating the most optimum test suite (most minimized test cases with covering all required combinations of parameters) is concerned. For this reason, metaheuristic AI algorithms which are based on natural and physics phenomena have been adopted for combinatorial software testing problem in the literature (for example, Particle Swarm Test Generator (PSTG) (Ahmed, Zamli, & Lim, 2012), Harmony Search (HS) (Alsewari, Poston, Zamli, Balfaqih, & Aloufi, 2020), Flower Pollination Algorithm (FPA) (Alsewari et al., 2018), Sine Cosine Algorithm (SCA) (Jalal M. Altmemi, Othman, & Ahmad, 2020) and Whale Optimization Algorithm (WOA) (Hassan, Abdullah, Zamli, & Razali, 2020), to name a few). Nonetheless, as generating test cases by t-way testing is considered as a Non-Deterministic Polynomial Time Hard Problem (NP-hard problem) (Kuliamin & Petukhov, 2011), no single t-way strategy can tackle all types of t-way configurations for the sake of optimality test suite generation. Thus, t-way testing is still considered open research problem and either introducing a new optimization algorithm or modifying the currently developed ones for optimal t-way test suite generation is encouraged (Bousono-Calzon, Bustarviejo-Munoz, Aceituno-Aceituno, & Escudero-Garzas, 2019).

Gravitational Search Algorithm (GSA) has demonstrated its effectiveness in a lot of research areas (Momeni, Yarivand, Dowlatshahi, & Armaghani, 2021)(Rashedi, Rashedi, & Nezamabadi-pour, 2018). The effectiveness of GSA is largely influenced by its gravitational attraction force that defines the algorithm's movement strategy of object

population (i.e., searcher agent population) in the search space (Rashedi, Nezamabadi-pour, & Saryazdi, 2009). The gravity mechanism in GSA well adjusts exploration and exploitation searches by causing objects with lighter masses accelerate towards the objects with heavier masses (i.e., directs to exploration or global search in search of global optimum). The slow movement of heavier objects triggered by the force ensures the exploitation or local search to look for optimal solutions around the neighbourhood of good solutions. In addition, GSA does not require to store the best position of each object obtained in each iteration which makes it feasible to solve large-scale optimization problems (i.e., problems that gets complexed exponentially according to the input size) (Philosophy, 2014).

Motivated by the afore-mentioned challenges and alluring prospects of GSA, this research thesis presents a new t-way test generation strategy known as Gravitational Search Test Suite Generator (GSTSG). In GSTSG, GSA is employed to generate t-way test suites. GSTSG is designed to focus on uniform strength t-way testing to provide optimal/near-optimal test suites compared to other existing strategies.

1.3 Aim and Objectives

This research aims to develop, implement and evaluate a new t-way test generation strategy, termed Gravitational Search Test Suite Generator (GSTSG), which focuses on minimizing test cases while covering required combinations of interaction elements at least once. To achieve the aim, the following objectives are set:

i. To develop a t-way test suite generator based on gravitational search algorithm (GSA)

ii. To evaluate the proposed strategy's performance (i.e., in terms of the size of generated test cases) by comparing with other existing t-way strategies

1.4 Research Scope

The latest Software Testing Lifecycle (STLC) defined by International Software Testing Qualifications Board (ISTQB) ("Certifying Software Testers Worldwide - ISTQB® International Software Testing Qualifications Board," n.d.) comprises five main stages: Planning, Analysis and Design, Implementation and Execution, Evaluating Exit Criteria and Documentation, as shown in Figure 1.2.

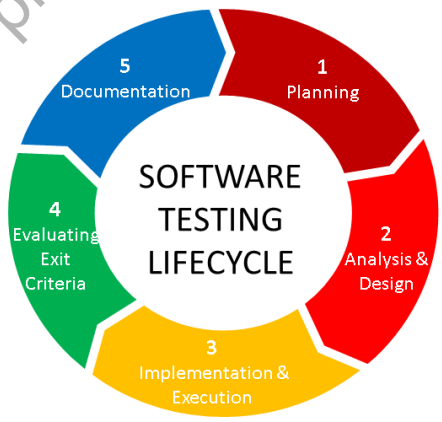

Figure 1.2 Stages of Software Testing Lifecycle (STLC)

The test planning stage involves considering the potential project timeline, such as determining tools and components to be utilized, resources (i.e., the number of test engineers needed), estimated costs, and scheduling the dates (i.e., when to start and stop

each project activity). During the analysis and design stage, the requirements document is analyzed, and the testing team creates a test design after the discussion with the software development team and project manager. The implementation and execution stage initializes with the preparation of the test environment, the establishment of expected test outcomes based on the previously created test design, and finally the execution of the developed system and the verification of whether the system behaves correctly when executing test cases from the test design during the execution process. The fourth stage of STLC evaluates when to stop or continue testing among dozens of test cases based on conditions, for instance, priority, risk, etc. Concerning the final documentation stage, all the detailed testing activities of the system under test (SUT) are archived in the documentation and sent to the customer service department for further actions.

This research mainly targets the test analysis and design stage, which is the generation of the test suite intending to develop a new t-way test suite generation strategy adopting a metaheuristic search technique, Gravitational Search Algorithm (GSA) called (Gravitational Search Test Suite Generator), GSTSG. The strategy constructs test suites using one-test-at-a-time (OTAT) method and can test any type of system configurations (i.e., SUTs) with uniform t-way interaction strength of up to t = 10. The test suite generated by this strategy is expected to be utilized in later stages of STLC.