A FUZZY ADAPTIVE TEACHING LEARNING-BASED OPTIMIZATION STRATEGY FOR GENERATING MIXED STRENGTH *T*-WAY TEST SUITES

FAKHRUD DIN

DOCTOR OF PHILOSOPHY

UNIVERSITI MALAYSIA PAHANG

UNIVERSITI MALAYSIA PAHANG

DECLARATION OF THESIS AND COPYRIGHT

Author's Full Name : FAKHRUD DIN

Date of Birth : APRIL 05, 1982

Title : A FUZZY ADAPTIVE TEACHING LEARNING-BASED
: OPTIMIZATION STRATEGY FOR GENERATING MIXED
: STRENGTH *T*-WAY TEST SUITES

Academic Session : SEM 2 2018/2019

I declare that this thesis is classified as:

☐ CONFIDENTIAL	(Contains confidential information under the Official Secret Act 1997)*
☐ 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 Universiti Malaysia Pahang reserves the following rights:

1. The Thesis is the Property of Universiti Malaysia Pahang
2. The Library of Universiti Malaysia Pahang has the right to make copies of the thesis 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)	(Supervisor's Signature)
BN7797082	Prof. Dr. Kamal Zuhairi Zamli
New IC/Passport Number	Name of Supervisor
Date:	Date:

SUPERVISOR'S DECLARATION

I hereby declare that I have checked this thesis and in my opinion, this thesis is adequate in terms of scope and quality for the award of the degree of Doctor of Philosophy.

_____

(Supervisor's Signature)

Full Name : PROF. DR. KAMAL ZUHAIRI BIN ZAMLI

Position : PROFESSOR

Date :

STUDENT'S DECLARATION

I hereby declare that the work in this thesis is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at Universiti Malaysia Pahang or any other institutions.

_____

(Student's Signature)

Full Name : FAKHRUD DIN

ID Number : PCC16010

Date :

A FUZZY ADAPTIVE TEACHING LEARNING-BASED OPTIMIZATION STRATEGY FOR GENERATING MIXED STRENGTH *T*-WAY TEST SUITES

FAKHRUD DIN

Thesis submitted in fulfillment of the requirements
for the award of the degree of
Doctor of Philosophy

Faculty of Computer Systems & Software Engineering

UNIVERSITI MALAYSIA PAHANG

April 2019

DEDICATION

Dedicated to my beloved parents and family.

For their countless prayers, endless love, unconditional support, tireless patience and continuous encouragement.

ACKNOWLEDGEMENTS

Thanks to All Mighty Allah, the Magnificent, the Most Merciful Who always bestows upon me His endless blessings. The completion of this thesis is yet another very special blessing that All Mighty Allah bestowed on me. Indeed, 'Then which of the favors of your Lord will you deny?'.

I am thankful to my supervisor, Professor Dr. Kamal Zuhairi Zamli, for his trust, patience and professional supervision. Without any doubt, his agile and goal-oriented supervision helped me to be an independent and productive researcher. I always got motivation and learned new research techniques whenever I met him. This work was only a dream without his novel style of supervision and dedication towards undertaking quality research. Thank you very much Prof! I consider myself very lucky to have you as my supervisor.

I am very grateful to my respectable parents for their kind prayers, selfless love and continuous support. I still remember my mother's sleepless nights she spent with me whenever I fell ill. I never have a hard time in my life because of my father's lifelong hard work. Khan G, you are my real hero! My parents are the best parents in the world. I am very thankful to my dearest wife for her unconditional support and company. I always feel blessed to have you in my life. Your presence and help made my Ph.D. journey very comfortable. Thank you very much, you are indeed a true, lovely and sincere partner. My special thanks go to my brothers and sisters, especially to my elder brother Mian Tufail Mohammad for his time and support for our family. I really love you all. I really appreciate the kind prayers of my mother-in-law and father-in-law. Finally, I would like to thank my entire family for their sincere prayers.

I would like to express my gratitude to my dear friends Captain Muhammad Sohail, Dr. Shah Khalid, Mr. Sami Ullah, Dr. Sami Ur Rahman, Mr. Mansoor Ahmed and Mr. Gohar Ali for their concerns and continuous support during my Ph.D. Thank you guys for being there for me always. I am very thankful to Mr. Riaz Ul Haq for his time and sincere help. I would like to thank Dr. Gran Badshah, Dr. Shahid Anwar, Dr. Mushtaq Ali and Mr. Wasif Nabeel Qureshi for their friendly time. Special thanks to my Ph.D. colleagues Dr. Hasneeza Lisa Zakaria and Dr. Abdullah for their help and time. I am grateful to Dr. Nomani Kabir for his kind help. I would like to give special thanks to Dr. Bestoun S. Ahmed and Dr. Mansoor for their valuable guidance and research tips. I am thankful to all my lab mates for their love and respect. I am grateful to all my colleagues especially to Dr. Nasir Rashid, Dr. Sehat Ullah, Dr. Aftab Alam, Mr. Anwar Ul Haq, Dr. Fakhre Alam and Dr. Zahid Khan from the Department of Computer Science & IT for their moral support. May Allah reward you and bless you all with the best health and more successes.

I express my gratitude to Ministry of Higher Education (MOHE), Malaysia for supporting my Ph.D. studies. Thank you MOHE for the all the support via the prestigious Malaysian International Scholarship (MIS). Also, I would like to thank MOHE for partially supporting my work and publications through FRGS grant entitled: A Reinforcement Learning Sine Cosine based Strategy for Combinatorial Test Suite Generation (RDU:170103). I am grateful to University Malaysia Pahang (UMP) for providing me the best infrastructure and conducive research environment and for supporting my studies. The time I spent at UMP is indeed the best time of my life which I will always remember. I would like to thank all faculty and staff members of Faculty of Computer Systems & Software Engineering (FSKKP) for their support and help. Last but not least, I am very grateful to the administration of University of Malakand for allowing me to pursue Ph.D. from Malaysia.

ABSTRAK

Penggunaan algoritma meta-heuristik sebagai asas untuk strategi *t*-cara (di mana *t* menunjukkan kekuatan interaksi) dan ujian kekuatan bercampur adalah perkara lumrah dalam kajian masa kini. Kebanyakan strategi penjanaan data ujian adalah berdasarkan algoritma meta-heuristik seperti Simulasi Penyepuhlindapan (SA), Pencarian Tabu (TS), Algoritma Genetik (GA), Pengoptimuman Koloni Semut (ACO), Pengoptimuman Gerombolan Zarah (PSO), Pencarian Harmoni (HS), Pencarian Burung Kedasih (CS), Algoritma Kelawar (BA) dan Algoritma Lebah yang telah dibangunkan pada tahun-tahun kebelakangan ini. Walaupun banyak kemajuan telah dicapai, penyelidikan ke atas strategi baru masih relevan kerana tiada strategi tunggal dapat mendominasi strategi sedia ada (seperti yang diramalkan oleh Teori Makan Tengahari Percuma). Di samping itu, kajian meta-heuristik bebas parameter tidak diterokai sepenuhnya dalam literatur saintifik. Oleh kerana prestasinya yang terbukti dalam banyak masalah pengoptimuman lain, penggunaan algoritma Pengoptimuman berasaskan Pembelajaran Pembelajaran (TLBO) yang bebas parameter sebagai strategi *t*-cara baru dirasakan amat berguna. Tidak seperti algoritma meta-heuristik yang sedia ada, TLBO adalah bersifat bebas parameter, dan tidak mempunyai sebarang kawalan parameter tertentu. Oleh itu, TLBO menghindarkan keperluan untuk proses penalaan khusus yang rumit dan tertumpu hanya pada bermasalah tertentu. Walau bagaimanapun, TLBO mengambil pendekatan yang mudah untuk melakukan carian global dan setempat secara berurutan pada setiap lelaran. Memandangkan proses eksplorasi (iaitu mencari lokasi baru yang berpotensi di ruang carian) dan eksploitasi (iaitu memanipulasi kejiranan setempat) adalah bersifat dinamik dan bergantung kepada ruang carian semasa, mana-mana pembahagian tetap antara keduanya boleh menjadikan proses carian kurang berkesan. Menangani isu-isu ini, tesis ini mencadangkan variasi TLBO baru berdasarkan sistem inferensi kabur Mamdani, yang dikenali sebagai Adaptif TLBO (ATLBO), untuk membolehkan pemilihan operasi carian global dan carian tempatan yang adaptif. Sistem inferensi kabur Mamdani mempunyai tiga masukan: pengukur kualiti, pengukur eksplorasi, pengukur eksploitasi dan satu keluaran pemilihan. Tiga masukan ini merekod keperluan bagi mencapai nilai optimum dengan membimbing prosess carian ke arah yang betul.Pengukuran kualiti dan explorasi digunakan untuk mencapai kepelbagaian penyelesaian, sedangkan langkah Intensifikasi digunakan untuk memudahkan penumpuan. Output sistem inferensi kabur Mamdani bertindak sebagai suis berselang-seli untuk pemilihan antara operasi carian global dan carian tempatan. Penerapan ATLBO untuk strategi penjanaan kekuatan ujian *t*-cara campuran menunjukkan prestasi yang kompetitif dari segi saiz sut ujian yang diperolehi berbanding TLBO asal dan algorithma meta-heuristik yang lain. Secara kesimpulannya, ATLBO menunjukan pencapaian secara purata terbaik sebanyak 39 untuk sut ujian yang dijalankan dengan mengunakan data eksperimen penanda aras dan merupakan strategi bebas parameter pertama boleh menghasilkan kedua-dua bentuk sut ujian iaitu keseragaman dan kekuatan bercampur parameter *t*-cara.

ABSTRACT

The use of meta-heuristic algorithms as the basis for *t*-way (where *t* indicates the interaction strength) and mixed strength testing strategies is common in recent literature. Many test data generation strategies based on meta-heuristic algorithms such as Simulated Annealing (SA), Tabu Search (TS), Genetic Algorithm (GA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), Harmony Search (HS), Cuckoo Search (CS), Bat Algorithm (BA) and Bees Algorithm have been developed in recent years. Although much progress has been achieved, research into new strategies is still relevant owing to the fact that no single strategy can claim dominance over other existing ones (i.e., as stipulated by the *No Free Lunch Theorem*). Additionally, the adoption of new *parameter-free* meta-heuristic-based *t*-way strategies has not been sufficiently explored in the scientific literature. Owing to its proven performance in many other optimization problems, the adoption of the *parameter-free* Teaching Learning-based Optimization (TLBO) algorithm as a new *t*-way strategy is deemed useful. Unlike most existing meta-heuristic algorithms, and by virtue of being *parameter-free*, TLBO does not have any specific parameter controls. Thus, TLBO avoids the need for cumbersome and problem specific tuning process. However, on the negative note, TLBO takes a simplistic approach of performing both global and local search sequentially per iteration. Given that exploration (i.e., globally finding the new potential region in the search space) and exploitation (i.e., locally manipulating best-known neighbourhood) are dynamic in nature depending on the current search space region, any preset division between the two can be counter-productive. Addressing these issues, this thesis proposes a new TLBO variant based on a Mamdani-type fuzzy inference system, called adaptive TLBO (ATLBO), to permit adaptive selection of its global and local search operations. The Mamdani-type fuzzy inference system of ATLBO has three inputs: Quality measure, Diversification measure and Intensification measure and one output: Selection. The three input measures capture necessary details so as to achieve optimality by guiding the search process in the right direction. Quality and Diversification measures are used to achieve solution diversity, whereas the Intensification measure is used to facilitate convergence. The Selection output of the Mamdani-type fuzzy inference system acts as an intermittent switch between global search and local search in ATLBO. The adoption of ATLBO for the mixed strength *t*-way test generation strategy demonstrates competitive performances in terms of obtained test suite sizes against the original TLBO and other meta-heuristic counterparts. To conclude, ATLBO-based strategy contributes to 39 new best average test suit sizes on benchmarking experiments and is the first *parameter-free* strategy that addresses generation for both uniform and mixed strength *t*-way test suites.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

$\sum$	Summation
!	Factorial
λ	Lamda
∃	There exists
+	Don't care
%	Percentage
P	Number of parameters
v	Values each parameter carries
t	Interaction strength
N	Number of rows
v^P	Number of parameters each carries v values
CA_i	i^{th} covering array
Cost(CA)	Cost of covering array
f(x)	Objective function value (total interaction elements covered)
x_i	Single interaction element
∈	Element of
H_s	Hash map of interaction elements
RM	Ringgit
/	Division
≥	Greater than or equal to
≤	Less than or equal to
⊇	Superset or equal to
A'	New solution
$f(A')$	Fitness function
p	Acceptance probability
r	Cooling rate
Δ	The difference in fitness functions
ε	Maximum evaluations
s_o	Initial solution
s^*	New solution
ρ	Neighborhood function
σ	Number of elite elements
Ƭ	Pheromone amount
Ŋ	Trial level

α	Pheromone coefficient
β	Heuristic coefficient
ρ	Pheromone evaporation rate
X_i	Candidate test case
gBest	Global best
lBest	Local best
C1, C2	Cognitive parameters
ω	Inertia weight
P_s	Combination list
V_i^t	Velocity of i^{th} particle at time t
r_{accept}	Acceptance rate
r_{pa}	Pitch adjustment rate
$\neg$	Not
p_a	Probability of finding cuckoo eggs in a nest
Q_i	Frequency of bat
n	Number of scout bees
m	Number of patches
e	Number of elite patches
Exp	Exponent
$C(c^*)$	Cost of new solution
C(c)	Cost of current solution
D	Dimension of the problem
X_i	Vector with D elements
$X_{teacher}$	Best learner in population X
X_{mean}	Mean of population X
$\neq$	Not equal to
X	Population
X′	Updated population
T_F	Teaching factor
X_{best}	Current best solution in population
I_m	Intensification measure
D_m	Diversification measure
Q_m	Quality measure
$X_{current}$	Current solution
$\mu(A(x))$	Membership function value of fuzzy set
max_fitness	Maximum fitness

min_fitness	Minimum fitness
—	Division
|	Such that
| |	Absolute value
∅	Empty set
t_{sub}	Sub strength
α	Significance level
α_{Holm}	Bonferroni-Holm correction
*	Best value
-	Result not available
∏	Product
f_i	Number of linguistic terms of the i^{th} linguistic variable

LIST OF ABBREVIATIONS

ATLBO	Adaptive Teaching Learning based Optimization
TLBO	Teaching Learning based Optimization
GA	Genetic Algorithm
SA	Simulated Annealing
PSO	Particle Swarm Optimization
ACO	Ant Colony Optimization
HS	Harmony Search
DPSO	Discrete Particle Swarm Optimization
APSO	Adaptive Particle Swarm Optimization
BA	Bat Algorithm
CS	Cuckoo Search
TS	Tabu Search
ACA	Ant Colony Algorithm
FPA	Flower Pollination Algorithm
PSTG	Particle Swarm-based Test Generator
CEG	Cause and Effect Graphing
ILS	Iterated Local Search
VNS	Variable Neighborhood Search
GLS	Guided Local Search
GRASP	Greedy Randomized Adaptive Search Procedure
CASA	Covering Arrays for Simulated Annealing
OA	Orthogonal Array
CA	Covering Array
MCA	Mixed Covering Array
VCA	Variable Strength Covering Array
CCA	Constrained Covering Array
SCA	Sequence Covering Array
CTCA	Cost-Aware Covering Array
OPAT	One-Parameter-At-A-Time
OSAT	One-Set-At-A-Time
OTAT	One-Test-At-A-Time
IEEE	Institute of Electronics & Electric Engineering
BSOD	Blue Screen of Dearth
SUT	Software/System Under Test

UML	Unified Modelling Language
GCC	GNU Compiler Collection
NP	Nondeterministic Polynomial time
CPHF	Covering Perfect Hash Families
MiTS	Mixed Tabu Search
TSA	Tabu Search Algorithm
GS	Genetic Strategy
VS-PSTG	Variable Strength Particle Swarm-based Test Generator
CPSO	Conventional Particle Swarm Optimization
HSS	Harmony Search-based Strategy
HM	Harmony Memory
HMS	Harmony Memory Size
BTS	Bat-inspired *t*-way Strategy
QOBL	Quasi Opposition Based Learning
FATLBO	Fuzzy Adaptive Teaching-Learning-based Optimization
MTLBO	Modified Teaching-Learning-based Optimization
ITLBO	Improved Teaching-Learning-based Optimization
DE	Differential Evolution
HSTLBO	Harmony Search-based Teaching-Learning-based Optimization
Co-TLBO	Cooperative Teaching-Learning-based Optimization
FL	Fuzzy Logic
COG	Centre of Gravity
FWER	Family-Wise Error Rate
GUIs	Graphical User Interfaces
ISA	Improved Simulated Annealing
SA-VNS	Simulated Annealing and Variable Neighborhood Search
ITL	Interaction Tuples List
IEL	Interaction Elements List
FTLS	Final t-way Test Suite
FTS	Final Sequence t-way Test Suite
IE	Interaction Element
TTLBO	Teacher's Teaching-learning-based Optimization

CHAPTER 1

INTRODUCTION

1.1 Overview

Errors in the life cycle of software are unavoidable. The action of any stakeholder involves in software development that leads to an unwanted result is called error. In the source code, design or resources of software, errors represent mistakes. When programmers commit mistakes while coding, they are known as bugs. The dynamic reliability of software such as correctness (the software expected behavior) is intensely affected by errors. Errors create defects, also known as faults. A faulty software component upon use causes software failures. A software system, from its requirements to maintenance, may either provide expected outcome (i.e., observing specifications) or encounter failures as shown in Figure 1.1. Errors and faults are associated with software artifacts such as use cases, unified modeling language (UML) models, hierarchy charts, etc. including source code. Failures happen to executable artifacts only, which is normally considered to be source code (Jorgensen 2016). The relationship between these terms is shown in Figure 1.2. When users encounter failure while using a software system or testers provoke failure during testing, it is reported as a problem or incident.

With the increasing complexity of today's software, the nature of faults has now become more challenging too. This is evident from the IEEE Standard Classification for Software Anomalies reported in (Zubrow 2009) that lists a range of faults such as input/output faults, logic faults, computation faults, interface faults, data faults, etc. (The document defines anomaly in software as "departure from the expected"). Large, configurable and other software applications, in addition to the above-mentioned faults, encounter a different kind of fault known as interaction fault. Such a fault can appear to be elusive. It may cause interaction failure when different software systems interact

(within an environment) or due to the combination of different features of same configurable software.

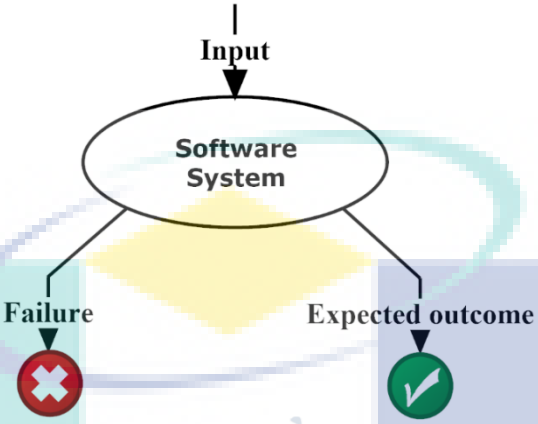


Figure 1.1 Failure and Software System

Interaction faults caused several Windows XP machines to crash. The incident resulted in the Blue Screen of Death (BSOD) problem and was reported in the Register news article (Leyden 2012). The problem occurred owing to the three-way interaction between Windows XP Cache manager, Symantec's security software and third-party encryption software. In GCC (GNU Compiler Collection), a framework for compilation, some interaction faults (configuration dependent) have been identified (Garvin and Cohen 2011). According to (Yin, Ma et al. 2011), 23.4%~61.2% faults in 5 software systems (COMP-A, CentOS, MySQL, Apache, and OpenLDAP) involved some combinations of parameters. Similarly, (Kuhn, Wallace et al. 2004) reported that in a variety of software systems faults could be triggered, though fewer, when three, four, five or six parameters interact. In essence, interaction faults can lead even to system crash owing to incompatibility problems of various factors or parameters of the system. (Niu, N et al. 2018). Therefore, exploring these faults is crucial for the success of a software system.

Interaction faults can be of two types: configuration faults and combination of parameter values faults. The former type may occur when various software systems interact. The latter one may arise when various parameters of the same software system are used in combination. Windows XP, Mozilla Firefox, and Apache Tomcat is a single configuration of an environment where three different software systems: Operating System, Web Browser and Web Server, respectively, are used. Similarly, Arial, Bold, 10

and Western is one combination of input values for the Font, Font Style, Font Size, and Script parameters in the Microsoft Notepad Font window.

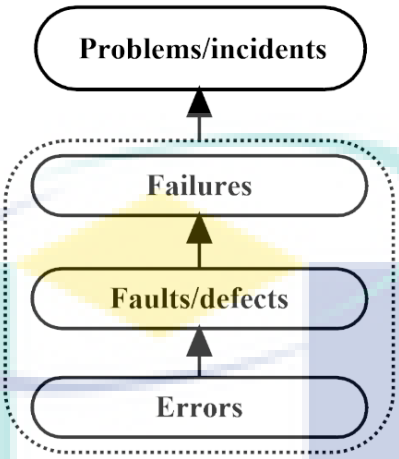


Figure 1.2 Relationship of Errors, Faults and Failures

It is common for today's software to have an exorbitant amount of combinations of input parameters or configurations. For example, the recent version of Apache server software has 172 input parameters. Of these, 158 are two-valued, 8 are three-valued, 4 are four-valued, 1 is five-valued and the final 1 is six-valued. This results in $1.8x10^{55}$ unique input combinations. Testing the web server exhaustively is infeasible even if only one second would reserve for each combination. It is even equally important to reduce testing efforts in case of software having a small number of input parameters.

Software testers wish to test Software Under Test (SUT) exhaustively. However, it is infeasible as almost every software today, like the Apache server software mentioned before, comes with dozens or even hundreds of input parameters. An appropriate solution point, known as combinatorial *t*-way (where *t* defines interaction strength) testing, is a collection of sampling strategies that efficiently sift out only selected input parameters sufficient for testing the large input space (Nie and Leung 2011). Combinatorial *t*-way testing represents a SUT as a model that contains the SUT's factors (configuration options or input parameters) each of which can be assigned values from a specific domain. This model is then used by a combinatorial *t*-way testing strategy to generate, for example, a combinatorial object called *t*-way covering array or simply covering array (CA). Specifically, a CA is a mathematical representation of a *t*-way test suite. In CA, each possible combination of input parameters values must appear at least once for each and every *t* (which is known as the interaction strength) input parameters' combinations. The

SUT is finally tested with CA by executing its test cases (i.e., each row in the CA) on each combination (Yilmaz, Fouch et al. 2014). An abstract level representation of combinatorial *t*-way testing is shown in Figure 1.3.

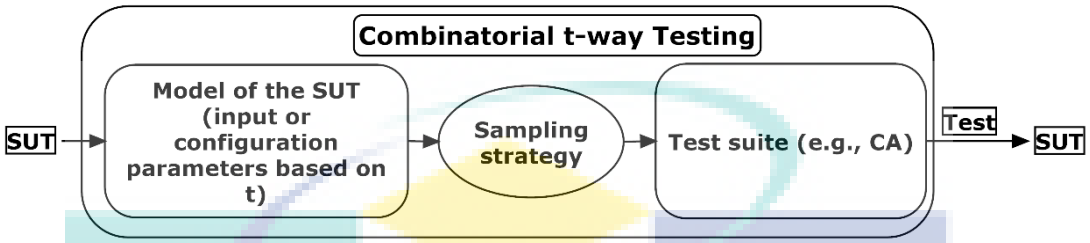


Figure 1.3 Abstract Level Representation of Combinatorial *t*-way Testing

In general, most *t*-way test suite generation strategies can be classified into three categories based on how CAs are generated:

i. One-test-at-a-time (OTAT): Strategies in this category repeatedly generate one test case as a single row of the CA until all required interactions are covered (Cohen, Dalal et al. 1997).

ii. One-set-at-a-time (OSAT): Strategies in this category generate a set of test cases at the end of each iteration. The strategy optimizes the coverage by mutating values of selected parameters of some test cases in the set. The size of the test set is decreased or even increased so as to achieve full coverage (Cohen, Colbourn et al. 2003, Nurmela 2004).

iii. One-parameter-at-a-time (OPAT): Strategies in this category initially do not generate complete test cases. Instead, they first assign values to some part of the input parameters to cover their interactions and subsequently set up the remaining part to generate complete test cases (Lei, Kacker et al. 2008).

The most flexible and efficient are the OTAT strategies as compared with the other two categories (Niu, N et al. 2018).

Generally, it is good enough to test only uniform interactions i.e., same interaction strength among all parameters of a SUT. However, only covering uniform interactions may not be sufficient in case of many contemporary real applications (Yilmaz, Cohen et al. 2006, Afzal, Torkar et al. 2009). For example, for some SUT there may be 100% two-

way interactions among all its parameters but may also be a 100% three-way (or higher) interactions among some subset(s) of its parameters. Such cases support the argument that while testing the SUT, the interaction strength might be variable instead of fixed. Therefore, *t*-way testing when *t* varies (variable or mixed strength interactions) be preferred than simple *t*-way testing (uniform strength interactions) owing to its flexible and practical nature (Nie and Leung 2011).

The problem of mixed strength test suites generation is a Nondeterministic Polynomial time (NP)-hard problem (Lei and Tai 1998). Significant research efforts have been made to investigate this problem. Recent efforts have focused on the adoption of meta-heuristic algorithms as the basis for the test suite generation strategies (Mahmoud and Ahmed 2015, Timaná-Peña, Cobos-Lozada et al. 2016) because these algorithms can achieve better results in terms of sizes compared with other computational strategies.

Meta-heuristic based strategies often start with a generation of random solution(s). One or more search operators are then iteratively applied to the solution(s) for improving the overall objective function evaluation (i.e., for greedy coverage of interaction combinations). Although several variations exist, the main difference among meta-heuristic strategies lies on each individual search operator and on the manipulation of exploration and exploitation. Owing to their success, many new *t*-way strategies based on meta-heuristic algorithms have been introduced in the literature.

Fuzzy control is widely adopted for tuning control parameters of meta-heuristic algorithms so as to balance exploration and exploitation. It is an active and useful research area that contributes to the applications of fuzzy logic and fuzzy sets (Castillo, Neyoy et al. 2015). Fuzzy controllers appear more effective in analyzing complex processes than conventional quantitative approaches. Similarly, the methodology followed by fuzzy controllers can be useful when the interpretation tools of available sources of information are qualitatively inaccurate or uncertain (Yen and Langari 1999).

1.2 Problem Statement

Software complexity and its operating environments may cause its behavior to be dependent on many factors. One important factor is unwanted interactions that can increase the occurrence of faults in software. Though useful, traditional software testing methods may not handle interaction faults owing to large input spaces of today's software

(Cohen, Dwyer et al. 2007). From last two decades, *t*-way test suite generation strategies effectively test software by sampling only required interactions. The *t*-way test suites generated successfully by these strategies cover interactions based on the required interaction strength at least once from typically a large number of parameter values or configuration options.

The problem of generating *t*-way test suites or mixed strength test suites is a computationally hard problem (Lei and Tai 1998, Yilmaz, Cohen et al. 2006, Afzal, Torkar et al. 2009, Kuliamin and Petukhov 2011). To be specific, searching for test suites with fewer possible test cases is an NP-hard problem (Lei, Kacker et al. 2008). Thus, it can be painstakingly difficult to search optimum mixed strength test suites. After being formulated as an optimization problem, several research studies emerged in the literature adopting meta-heuristic algorithms for the generation of near-optimal test suites. Meta-heuristic algorithms adopted by *t*-way test suite generation strategies include Simulated Annealing (SA) (Cohen, Colbourn et al. 2003), Tabu Search (TS) (Nurmela 2004), Genetic Algorithm (GA) (Shiba, Tsuchiya et al. 2004), Ant Colony Algorithm (ACA) (Shiba, Tsuchiya et al. 2004), Particle Swarm Optimization (PSO) (Ahmed and Zamli 2011c, Ahmed, Zamli et al. 2012), Harmony Search (HS) (Alsewari and Zamli 2012), Cuckoo Search (CS) (Ahmed, Abdulsamad et al. 2015), Bat algorithm (BA) (Alsariera and Zamli 2015), and Bees Algorithm (Mohd Hazli, Zamli et al. 2012). Although much progress has been achieved (Timaná-Peña, Cobos-Lozada et al. 2016), research into new strategies is still relevant owing to the fact that no single strategy can claim dominance over other existing ones (i.e., as stipulated by the *No Free Lunch Theorem* (Wolpert and Macready 1997)).

Most meta-heuristic algorithms introduce specific parameter controls so as to search optimum solutions (i.e., *t*-way test suites). For example, GA exploits crossover probability, mutation probability, selection operator, etc.; PSO introduces inertia weight and social/cognitive parameters; HS relies on the consideration rate of harmony memory and pitch adjustment; ACO exploits evaporation rate, pheromone influence, and heuristic influence; SA uses temperature and cooling rate; TS introduces short-term memory and long-term memory, CS relies on switching probability; BA exploits frequency, loudness and pulse emission rates; and Bees Algorithm uses the number of scout bees, the number of patches, the number of elite patches, etc. Tuning such control parameters accordingly

ensures a suitable quality solution. However, the tuning of these parameters is often time consuming and problem specific because there is no one-size-fits-all strategy. Therefore, it is necessary to design new *t*-way strategies based on *parameter-free* meta-heuristic algorithms as the adoption of the same has not been explored in the scientific literature as depicted in Figure 1.4.

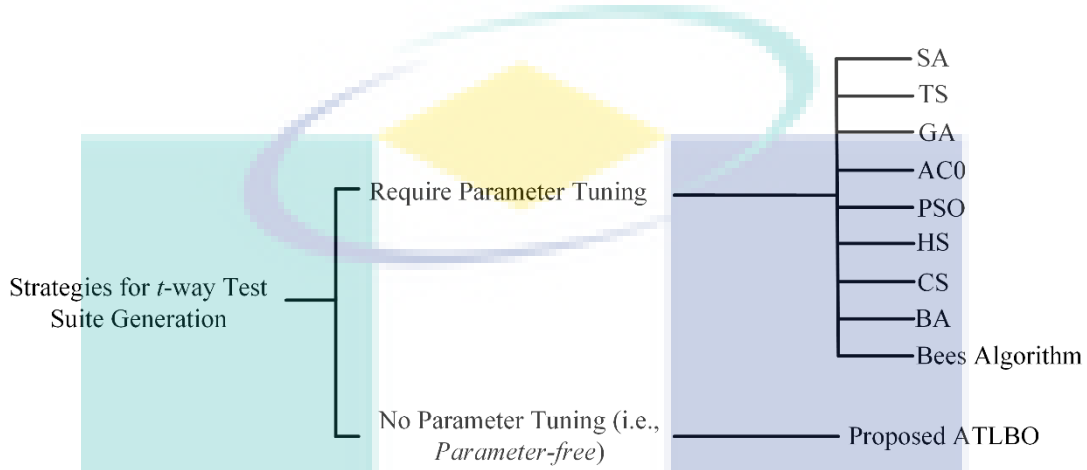


Figure 1.4 Strategies with/without Parameter Tuning for the Problem of *t*-way Test Suite Generation

Owing to its proven performance in many other optimization problems, the adoption of the *parameter-free* Teaching Learning-based Optimization (TLBO) (Rao, Savsani et al. 2011, Rao, Savsani et al. 2012) algorithm as a basis for a new *t*-way strategy is deemed useful. Unlike most existing meta-heuristic algorithms, and by virtue of being *parameter-free*, TLBO does not have any specific parameter controls. Thus, TLBO avoids the need for cumbersome and problem specific tuning process. However, on the negative note, TLBO takes a simplistic approach of performing both global search and local search sequentially per iteration. Given that exploration (i.e., globally finding new potential region in the search space) and exploitation (i.e., locally manipulating best-known neighborhood) are dynamic in nature depending on the current search space region, any preset division between the two or their 50-50 probability as in TLBO can be counter-productive (M. Črepinšek, Liu et al. 2013, Yang, Deb et al. 2013). For instance, Figure 1.5 (a) shows that the search needs to be intensified as convergence to the global minimum is near. Local search may complete its turn at this point of the search process and is immediately followed by global search. This causes TLBO to miss the global minimum as shown in Figure 1.5 (b). Similarly, Figure 1.5 (c) requires more exploration than exploitation as the illustrated problem has more than one optimal solutions (i.e.,

multimodal). On the other hand, exploitation is required in case of the search space for a unimodal problem depicted in Figure 1.5 (d).

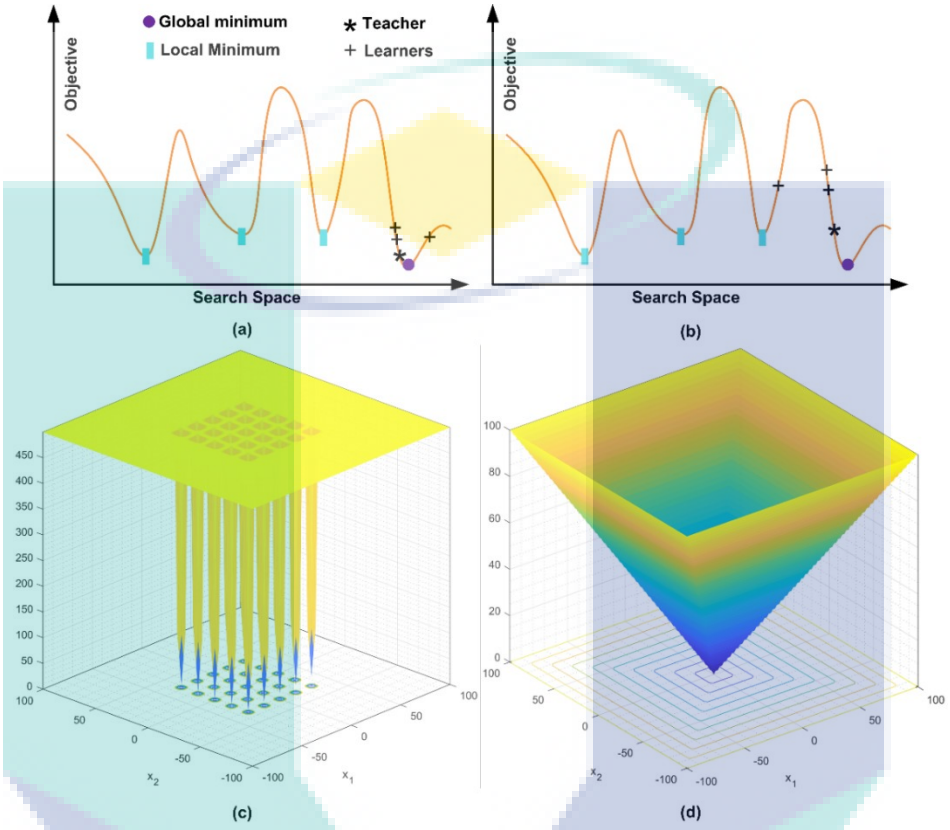


Figure 1.5 Performance Issues with the Original TLBO Algorithm

To address these issues, this thesis presents a new TLBO variant called adaptive TLBO (ATLBO) integrated with the Mamdani-type fuzzy inference system (Cordón 2011, Camastra, Ciaramella et al. 2015) for the problem of generating mixed strength *t*-way test suites. ATLBO adaptively selects either local search operation or global search operation per iteration. This new capability enables ATLBO to have a good balance between exploration and exploitation. Experimental results reveal that ATLBO exhibits competitive performances in terms of obtained mixed strength test suite sizes against original TLBO and other meta-heuristic counterparts. In essence, the hypothesis of this thesis suggests that ATLBO is very effective for the problem of generating mixed strength *t*-way test suites owing to its improved search mechanism and a good balance between global search and local search.

1.3 Aim and Objectives

The quest for new or enhanced meta-heuristic algorithms to effectively and efficiently solve the mixed strength *t*-way test suite generation problem is the main motivation of this research work. Therefore, this thesis aims to propose an enhanced TLBO variant called adaptive TLBO (ATLBO) using a Mamdani-type fuzzy inference system for the mixed strength test suite generation problem. For fulfilling this aim, the objectives of the research are:

i. To design a new variant of TLBO called ATLBO based on a Mamdani-type fuzzy inference system for adaptively selecting exploration (i.e., global search) and exploitation (i.e., local search).

ii. To implement ATLBO for addressing generation for both uniform and mixed strength *t*-way test suites.

iii. To evaluate the performance of ATLBO in terms of generated test suite sizes against the original TLBO and other meta-heuristic algorithms.

1.4 Research Scope

Following points highlight the scope of this research work.

- The focus of this thesis is the test case generation phase in the software testing life cycle. Specifically, the mixed strength *t*-way test suite generation/sampling for test execution is addressed in this research work.

- The methodology adopted by a *t*-way strategy can either be exact, algebraic, greedy, or meta-heuristic. Strategies based on meta-heuristic algorithms have generated most optimal test suites to date. Therefore, only meta-heuristic-based strategies for *t*-way testing are considered in this thesis.

- Meta-heuristic algorithms with no algorithm-specific parameters (i.e., *parameter-free*) have been recently introduced in the optimization literature. Besides TLBO, Jaya (Rao 2016) and Symbiotic Organisms Search (SOS) (Cheng and Prayogo 2014) are some other examples of

parameter-free meta-heuristic algorithms. This research has adopted TLBO owing to its successful applications for solving a wide range of optimization problems in many fields of science and engineering.

- The scope of this thesis is limited to the design and implementation of a mixed strength *t*-way test suite generation strategy based on ATLBO. The current interaction strength support is set at $t = 4$ which is empirically consistent with evidence in the literature (where 70-90% of the faults can be detected).

- Two main types of fuzzy inference systems, namely Mamdani (Mamdani and Assilian 1975) and Sugeno (Sugeno 1985) have broad acceptance and applications for solving many real-world problems. This work integrates the well-known Mamdani-type fuzzy inference system with ATLBO so as to achieve a suitable performance. Sugeno as well as other types of fuzzy inference systems are outside the scope of this thesis.

- This thesis focuses only on the test generation and not on the test execution. As a result, the performance of ATLBO for the mixed strength *t*-way test suite generation is gauged in terms of obtaining the most minimum test suite size.

1.5 Research Activities

Overall, the research activities encompass three main phases, namely the literature review, methodology and evaluation. The first phase is the literature review that leads to a complete understanding of the domain of interest via the available state-of-the-art literature. The research problem and proposed research contribution are formulated from this phase. The second phase constitutes the methodology of this work where initially the general ATLBO with its new search mechanism is presented followed by giving the complete details of adopting ATLBO for addressing generation for both uniform and mixed strength t-way test suites. Finally, the evaluation/benchmarking phase investigates the performance of ATLBO in terms of the generated test suite sizes against referenced strategies based on original TLBO and other well-established meta-heuristic algorithms. To illustrate how the phases are related, the research activities are summarized in Figure 1.6. The following subsections elaborate these phases further.

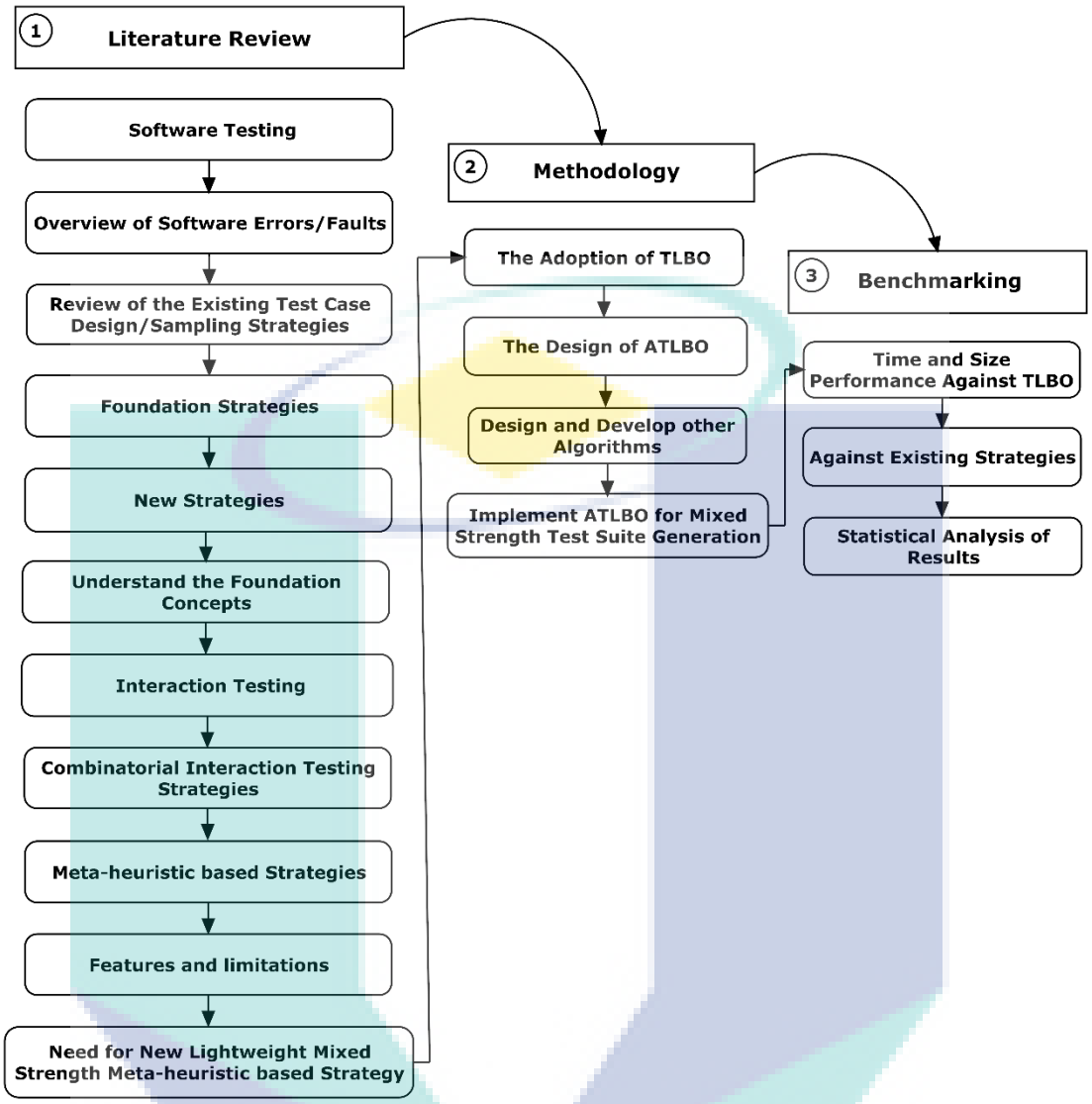


Figure 1.6 Research Activities

1.5.1 Literature Review

In this phase, the literature survey is carried out to understand the current state of research on the uniform and mixed strength *t*-way testing. Software testing is considered as general research area which is then narrowed down to test data generation strategies owing to their alluring prospects. The literature review starts by reviewing existing foundation test case design strategies along with newly establishing strategies for exploring software interaction faults. After identifying the importance of the complementary test case design strategies, existing strategies in the literature for *t*-way testing are reviewed in order to identify their features and possible limitations. The research statement is formulated after the literature review survey. This phase provided the foundation for the methodology of this research study.

1.5.2 Methodology

In this phase, TLBO is adopted owing to its simple design for optimization. A new variant of TLBO, called ATLBO is designed that intelligently applies the local and global search operations using a Mamdani-type fuzzy inference system. After presenting the general structure of ATLBO, the details of its implementation for addressing the problem of mixed strength *t*-way test suite generation are given. Other more optimized supporting algorithms are also designed and developed in this phase. Detailed methodology of this research study for addressing the problem of mixed strength *t*-way test suites will be discussed in Chapter 3.

1.5.3 Benchmarking

Initially, ATLBO is benchmarked against the original TLBO for a predefined set of example systems to evaluate its performance (in terms of time) and efficiency (in terms of generated test suite sizes). Then, the results of ATLBO are benchmarked against the well-known results of other state-of-the-art meta-heuristic based strategies to further investigate and evaluate its efficiency.

1.6 Thesis Structure

This thesis is organized into five chapters. The organization outlined in this section is as follows. Chapter 2 reviews the foundation knowledge relevant to software interaction testing and meta-heuristic based test case generation strategies. The chapter starts with the introduction of basic test case design strategies with examples that illustrate how each strategy selects test cases. Then, a simple online gaming architecture is presented as a configuration software system to illustrate uniform and mixed strength interaction testing. Thereafter, the chapter explains the formulation for interaction elements and how these elements are covered by test cases. This is followed by a theoretical background of *t*-way testing. The chapter then overviews meta-heuristic algorithms. Then, it briefly describes and investigates uniform and mixed strength interaction test suite generation strategies based on meta-heuristic algorithms such as SA, TS, GA, ACA, PSO, HS, CS, BA and Bees Algorithm available in the existing literature. Here, the search process and algorithm structure of these meta-heuristic based strategies are described. Following this, strategies based on meta-heuristic algorithms for *t*-way testing are categorized into standard, adaptive and hybrid strategies. The chapter then

presents an overview of TLBO and reviews its different variants with their applications. Hereafter, the chapter identifies the research gap in the application of these meta-heuristic-based strategies for *t*-way testing.

Chapter 3 presents the methodology that justifies how ATLBO is designed and implemented for addressing the *t*-way test suite generation problem. After the overview of original TLBO, the detailed design of ATLBO is given. Specifically, the Mamdani-type fuzzy inference system is briefly discussed which ATLBO uses for the global search and local search selection as per search requirement. Finally, the implementation of ATLBO for mixed strength *t*-way test suites is presented with thorough explanations on how it can be used for automated test data generation.

Chapter 4 evaluates performance of ATLBO against other strategies based on meta-heuristic algorithms in terms of the generated test suite sizes. ATLBO's performance is first evaluated in terms of both time and generated test suite size against original TLBO. Similarly, the generated test suite sizes by ATLBO are benchmarked against the results available in the high-impact literature. All experimental results are reported in the form of best and mean solutions. Similarly, the chapter also includes statistical analysis conducted for all the obtained results to ensure better comparison. For every test generation problem, the chapter depicts the percentage distribution pattern of exploration and exploitation adopted by ATLBO. The chapter then briefly discusses the observations based on the obtained results. At the end, the chapter presents the identified threats to validities and elaborates how to mitigate their effects on the results.

Finally, chapter 5 concludes this research study with a summary of achievements and contributions. Moreover, it revisits the main research hypothesis and debates on the effectiveness of ATLBO. Eventually, the chapter derives conclusions based on conducting this research and its findings and presents future work.

CHAPTER 2

LITERATURE REVIEW

The foundation for this work has been laid out in chapter 1. The area of interest has been elaborated in the chapter followed by a brief description of the problem statement. The aim and the objectives that drive this study have been stated. Finally, the structure of the thesis has given in the chapter.

Complementing chapter 1, this chapter reviews the body of knowledge in the area of automatic test case design to understand it, and subsequently identifies the research problem. Firstly, the chapter explains the basic test case design techniques. Secondly, the chapter presents the fundamentals of combinatorial mixed strength and *t*-way testing. Thirdly, the chapter comprehensively reviews the state-of-the-art meta-heuristic-based *t*-way strategies along with their novel division into three categories, namely standard, adaptive and hybrid strategies. Fourthly, the chapter presents an overview of original TLBO and briefly reviews its variants and their applications. Fifthly, the chapter identifies the research gap in the existing works of the application of meta-heuristic algorithms for *t*-way test suite generation. Lastly, the chapter concludes the presented contents in the summary section.

2.1 Test Case Design Techniques

Effective testing is based on efficient test case design techniques. These techniques enable testers to select test cases that best suit the system. Some of the well-known test case design techniques are explained with examples in the following subsections.

2.1.1 Equivalence Partitioning

Equivalence partitioning tests the Software/System Under Test (SUT) with equally partitioned classes of its input/output domains (Hass 2014). With such a technique, testing the SUT using a selected test case in a class is sufficient to test it with all other possible test cases in that class partition. To put it differently, the value of any test case in a class is supposedly equivalent to the value of any other test case in that class partition. Thus, if a test case is successful in fault detection in a class, it is equivalently attributed to the other test cases as well in the same class partition (Myers, Sandler et al. 2011). To clearly illustrate this concept, a simple example is given in Figure 2.1.

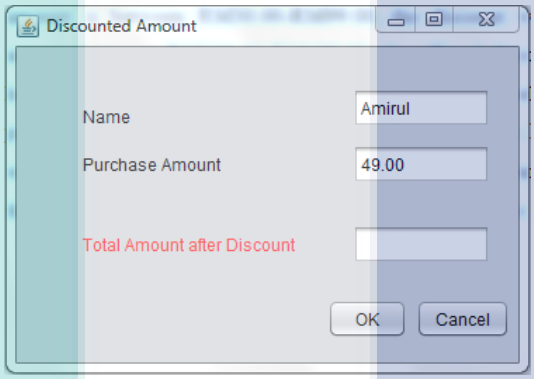


Figure 2.1 A Simple Application to Illustrate Equivalence Partitioning

A simple application that calculates the total amount of a customer bill after discount based on the purchase amount is presented. For example, if the purchase amount is between RM30.00-RM99.00, the discount will be 5%. For the purchase amount above RM99.00, the offered discount is 10%. It can be observed here that there is no discount for the purchase amount below RM30.00. Testing all the possible values of the purchase amount field is infeasible. Hence, the values of the purchase amount can be partitioned into classes using equivalence partitioning that determine the discount. Clearly, in case of the running example, there are three classes of the purchase amount, i.e., purchase amount below RM30.00, purchase amount above RM99.00, and purchase amount between RM30.00-RM99.00. To test the Amount after Discount field of the application, only one value (possibly the value at the mid) needs to be selected from each class partition. Therefore, 14.00, 49.00, and 200.00 are required to test the system using equivalence partitioning.

2.1.2 Boundary Value Analysis

Boundary value analysis uses the class partitions of equivalence partitioning for the selection of test cases. Test cases selected using boundary value analysis include the values exactly on, below and above the edges of the class partitions as many faults could also occur using such values (Burnstein 2006, Myers, Sandler et al. 2011).

In the case of example depicted in Figure 2.1, the boundaries of the three class partitions are 1.00, 29.00, 99.00 and 100.00, respectively. To undertake the boundary value analysis, testers for this example would use (0.00, 1.00, 2.00), (28.00, 29.00, 30.00) and (99.00, 100.00, 101.00) as test cases.

2.1.3 Cause and Effect Graphing

Cause and Effect Graphing (CEG) is another specification-based test design technique for validating the functionality of a given SUT. CEG is more effective for control centric SUTs as compared to the previously discussed techniques which are used with data-centric SUTs (Srivastava, Patel et al. 2009). By using a graph, CEG visualizes the SUT's inputs (or causes) with their corresponding outputs (or effects) (Hass 2014). The cause corresponds to an input condition from the specification that may influence the result of the SUT, whereas the effect corresponds to the response of the SUT to any set of input conditions (Srivastava, Patel et al. 2009).

Initially causes, effects and constraints are identified from the SUT's specifications when tests adopt CEG. This is followed by the construction of cause and effect graph as a combinatorial logic network graph. The nodes of the graph represent causes, effects and constraints whereas its edges represent Boolean operators (AND, OR, NOT) between causes and effects. After the graph construction, a unique identifier is assigned to each cause and effect and their relationships are marked on the graph. The next step transforms the graph into a decision table to design test cases.

For the example application shown in Figure 2.1, suppose that the SUT's specification requires that the Purchase Amount field needs not be negative and that its value should not contain any alphabets or special symbols. The SUT should display an error message if the value of Purchase Amount field is negative, and if it contains alphabet(s) or special symbol(s); the "Invalid value" message needs to appear. Otherwise,

the Total Amount after Discount value should appear. In this case, the input conditions or causes are C1: The value of Purchase Amount > 0.00 and C2: The value without alphabets or special symbols. Similarly, the output conditions or effects are E1: The value is negative; E2: The Total Amount after Discount value and E3: An Invalid value. CEG visualizes the relationships between these causes and effects in Figure 2.2.

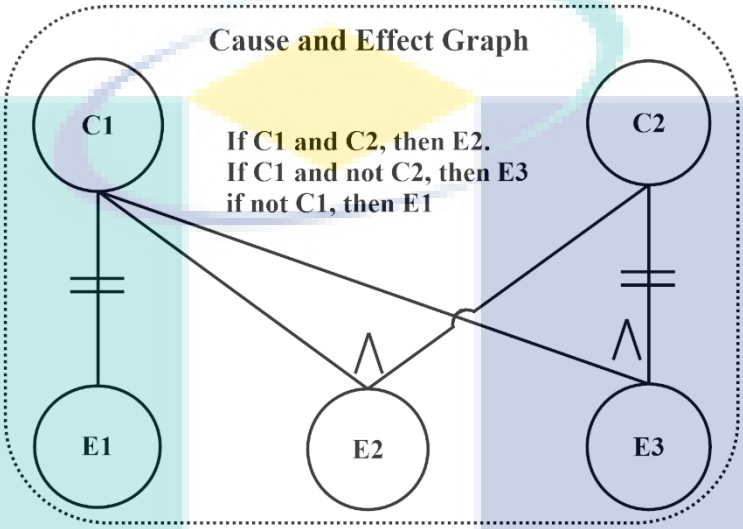


Figure 2.2 The CEG for the Example in Figure 2.1

In the decision table, shown in Table 2.1, rows represent causes and effects whereas columns represent test cases (three test cases T1, T2 and T3 for the running example). The entry of each cell of the table may be '0', '1', or '+'. If a cell entry is '0', it indicates the absence of cause or effect. If a cell entry is '1', it indicates the presence of cause or effect. The entry '+' indicates 'don't care'.

Table 2.1 Decision Table for the Example in Figure 2.2

causes / effects	Test cases		
	T1	T2	T3
C1	1	1	0
C2	1	0	+
E1	0	0	1
E2	1	0	0
E3	0	1	0

The above-discussed techniques are useful to discover and prevent faults. However, such techniques are unable to detect faults due to interactions of input

components (Cohen, Dwyer et al. 2007). Combinatorial *t*-way testing among other techniques is found most effective to avoid the otherwise impossible exhaustive testing.

2.2 Theoretical Background: Mixed Strength and *t*-way Testing

It has been proved empirically that only a small number of inputs (usually from 3 to 6) causes faults in certain classes of software (Kuhn and Okum 2006). If *t* or fewer input parameters involved in the occurrence of a fault, a smaller set of test cases can be designed on some *t*-way combinations. These test cases appear to be very effective as they can detect 50% to 75% of the faults in a SUT (Kuhn, Wallace et al. 2004).

2.2.1 Mixed Strength and *t*-way Testing: A Motivating Example

To illustrate the generation problem of both mixed strength and *t*-way test suites, a simple model of online gaming architecture is presented as shown in Figure 2.3.

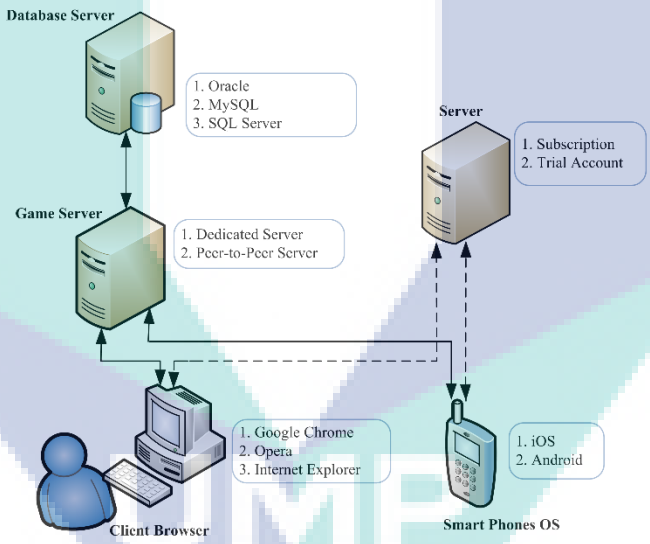
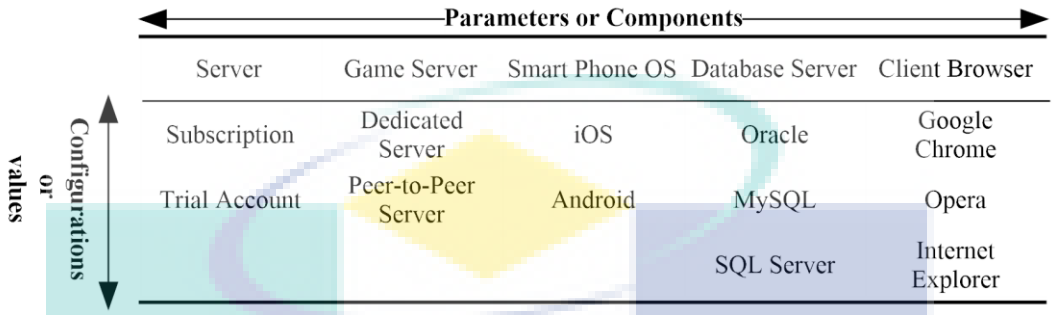


Figure 2.3 Online Gaming Architecture

The online gaming architecture is composed of five parameters or components. There are two types of clients (i.e., parameters are 'Client Browser' and 'Smart Phone OS') where parameter 'Client Browser' represents a normal PC user whereas parameter 'Smart Phone OS' represents a mobile user. Both parameters have different configurations or values. On the server side, there are three types of parameters, namely 'Server', 'Game Server' and 'Database Server'. Each of these parameters carries different values. Thus, the online gaming architecture depicted in Figure 2.3 can be summarized

(see Table 2.2) as a system of five parameters where three parameters carry two values while two parameters carry three values.

Table 2.2 The Online Gaming Architecture: Parameters and Values

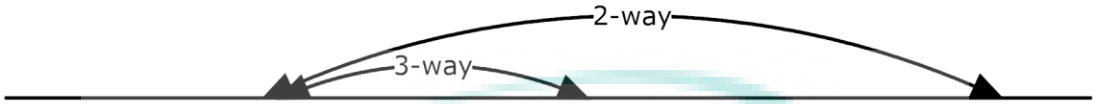


This system can be tested with different testing methods, but one of the common sources of software faults may be some unexpected interaction between the system's parameters or configurations (Williams and Probert 2001). The chances of failure of a system increase with the increase in number of parameters. To mitigate such chances, and ensure the quality of a SUT, testers may require to perform exhaustive testing (i.e., test all combinations or interactions among parameters). For the system discussed here, exhaustive testing yields a total 72 test cases (i.e., 2x2x2x3x3). However, there can only be 9 test cases to execute if *two*-way interactions of the system parameters are considered as shown in Table 2.3 (first 9 test cases). Though this will give a minimum number of test cases and will cover all the *two*-way combinations but may miss some interaction faults. For example, the parameters; Server, Game Server, and Smart Phone OS need to be tested collectively to avoid possible faults because of their interactions. One way of achieving this is to first prepare *two*-way test cases for all five parameters followed by preparing *three*-way test cases for the three stated parameters. This will result in 21 test cases. However, the testing cost will be increased due to a large number of test cases, particularly in cases where the system is highly configurable or has large input parameters.

Another way of designing test cases for *two*- and *three*-way interactions simultaneously is combining them in a single test suite. This method not only maintains minimal coverage across the parameters but also avoids testing all the 71 or 21 test cases. Table 2.3 shows the test suite that covers the variable or mixed strength interactions among the parameters of the system shown in Figure 2.3. There are now only 10 test

cases to cover pairwise or *two*-way interactions of all the five parameters of the system and *three*-way interactions of the three parameters mentioned above.

Table 2.3 Pairwise and Mixed Strength Test Suite for the System in Figure 2.3



Test case #	Server	Game Server	Smart Phone OS	Database Server	Client Browser
1	Trial Account	Dedicated Server	iOS	MySQL	Google Chrome
2	Subscription	Dedicated Server	Android	SQL Server	Internet Explorer
3	Subscription	Peer-to-Peer Server	iOS	Oracle	Opera
4	Trial Account	Peer-to-Peer Server	Android	Oracle	Google Chrome
5	Trial Account	Peer-to-Peer Server	iOS	SQL Server	Internet Explorer
6	Subscription	Peer-to-Peer Server	Android	MySQL	Opera
7	Trial Account	Dedicated	Android	SQL Server	Opera
8	Subscription	Dedicated Server	iOS	SQL Server	Google Chrome
9	Trial Account	Dedicated Server	iOS	Oracle	Internet Explorer
10	Trial Account	Dedicated Server	iOS	MySQL	Internet Explorer

The above discussion signifies the importance of defining and creating test suites that cover both *t*-way and mixed strength interactions, particularly when interactions grow. To save testing cost and time, it is unavoidable to search for effective and efficient ways for creating test suites with minimum possible test cases, specifically for systems with large configurations or with large number of input parameter values.

2.2.2 Basics of Interaction Coverage

In this section, a simplified example is presented to comprehend the process of test suites creation and interaction elements coverage. In Table 2.4, a system with three parameters (*a*, *b*, *c*) each with two values is presented.

Table 2.4 A Simplified Example of a System with three Parameters two Values

	Parameters →		
	a	**b**	**c**
Values ↕	a_1	b_1	c_1
	a_2	b_2	c_2

For this example, as illustrated in the previous section, the test suite with total 8 (i.e., 2x2x2 or 2^3) test cases as shown in Table 2.5 can test the system exhaustively (i.e., at full strength $t = 3$).

Table 2.5 The Exhaustive Test Suite for the System in Table 2.4

	Test Case #	**a**	**b**	**c**
Test Suite ↕	1	a_1	b_1	c_1
	2	a_1	b_1	c_2
	3	a_1	b_2	c_1
	4	a_1	b_2	c_2
	5	a_2	b_1	c_1
	6	a_2	b_1	c_2
	7	a_2	b_2	c_1
	8	a_2	b_2	c_2

With the increase in number of parameters and values, the size of exhaustive test suites increases exponentially. Pairwise test suites are helpful as they significantly reduce the number of test cases and achieve complete coverage of the interaction elements or tuples. Mathematically, the total number of these interaction tuples can be calculated exactly using Eq. 2.1 (Colbourn and Dinitz 2006).

$$\text{Total Interaction Tuples or Elements} = \binom{P}{t} v^t = \frac{P!}{t!(P-t)!} v^t \qquad 2.1$$

Eq. 2.1 calculates the total number of interaction tuples using the number of system's parameters *(P)*, the required interaction strength *(t)* and the number of values *(v)* each parameter carries. For cases where P, t, and v vary, the total number of interaction elements is equal to the sum of products of each individual interaction sets. The first step that any t-way testing strategy takes is to determine all the possible combinations of the system parameters or interaction sets using Eq. 2.2.

$$\text{Total Parameters Combinations} = \binom{P}{t} = \frac{P!}{t!(P-t)!} \qquad 2.2$$

For $P = 3$ and $t = 2$, Eq. 2.2 results in the value three. Therefore, for the example in Table 2.4, there are three possible parameter combinations, namely *ab*, *ac* and *bc*.

After this calculation, the *t*-way strategy assigns related values to every combination of the parameters to form interaction elements or tuples. The strategy then attempts to generate test suite with fewer possible test cases in order to cover all these tuples. Figure 2.4 shows parameters combinations (total three) and interaction elements (total 12) for the system in Table 2.4.

Combinations of System Parameters

Tuple #	a	b	c	Tuple #	a	b	c	Tuple #	a	b	c
1	a_1	b_1	-	5	a_1	-	c_1	9	-	b_1	c_1
2	a_1	b_2	-	6	a_1	-	c_2	10	-	b_1	c_2
3	a_2	b_1	-	7	a_2	-	c_1	11	-	b_2	c_1
4	a_2	b_2	-	8	a_2	-	c_2	12	-	b_2	c_2

Figure 2.4 Total Pairwise Interaction Tuples for the System in Table 2.4

The final test suite covers all the interaction tuples at least once to ensure that all tuples are tested at least once. The process of constructing test cases demonstrates the coverage of interaction elements. A test case, based on its arrangement, could be sufficient for testing either one or more interaction elements. If a test case covers more elements, it is the more suitable candidate to be added to the final test suite. Figure 2.5 illustrates how only four test cases (a reduction of 50% testing efforts) cover all the 12 interaction tuples shown in Figure 2.4.

Although, 100% coverage is achieved with only four test cases for the running example, the same percentage of tuples can be covered by more than four test cases if other test cases were chosen instead of those shown in Figure 2.5. The first test case (a_1, b_1, c_1) in Figure 2.5 achieves 25% coverage of the total pairwise interaction tuples as it covers three interactions. With the addition of the second test case i.e., (a_1, b_2, c_2), coverage percentage reaches 50 as three more interactions have been covered now. The addition of the last two test cases in the test suite completes 100% coverage as they cover

the remaining six interactions. Interaction elements are removed once they are covered by a test case.

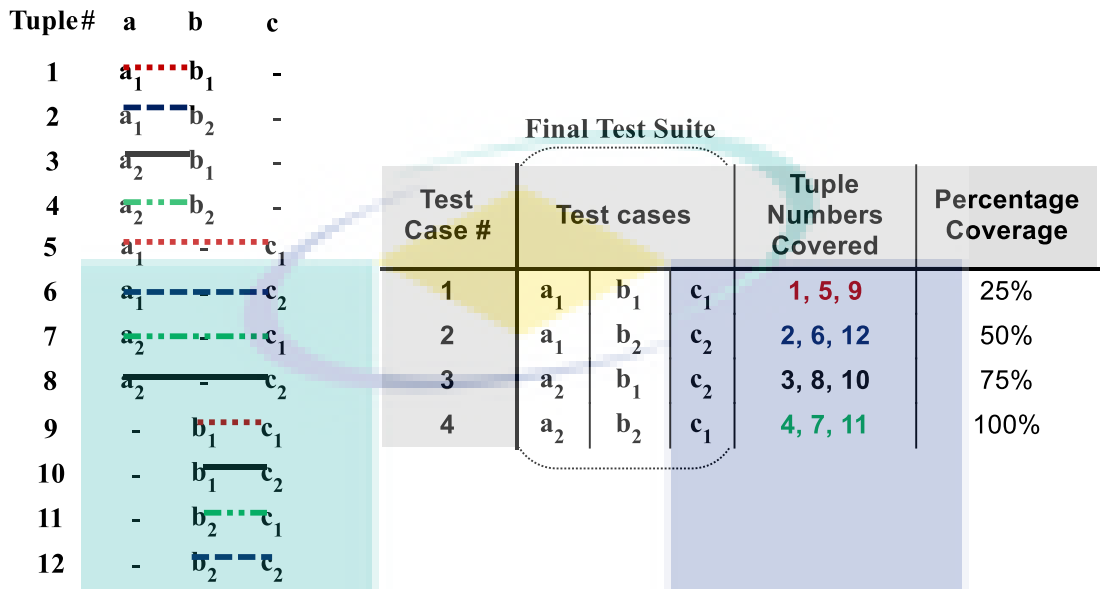


Figure 2.5 Interaction Elements/Tuples Coverage for the System in Table 2.4

The process of interaction coverage explained above is usually followed by most of the greedy *t*-way test suite generation strategies. However, test case selection and test suite generation differentiate one *t*-way strategy from another.

As the problem of *t*-way test generation is NP-hard, a strategy can only predict an approximate number of test cases using the lower bound. For uniform *t*-way test suites, the lower bound is computed as the interaction strength times values (i.e., v^t). When the test suite size reaches the lower bound, it is called an absolute minimal. A strategy cannot produce test suite size minimum than the lower bound. In a SUT with different *P* and *v* while only a single interaction strength *t*, the lower bound here can be computed by multiplying interaction strength *t* times the first maximum values of the parameters in descending order. For mixed strength *t*-way test suites, the lower bound is the sum of products of the first maximum values in descending order times the main and sub interaction strengths.

2.2.3 Mathematical Objects for Test Suites Representation

There are seven different mathematical objects reported in the literature for the mathematical representation of *t*-way test suites (See Table 2.6). Covering array (CA) is

the most common type of mathematical object used by strategies in *t*-way test suite generation. CAs were initially used for interaction testing by Williams and Probert (Williams and Probert 1996) and Cohen et al. (Cohen, Dalal et al. 1997). With CAs, the input space and its domain of values (or configurations options) of a software system can be modeled efficiently. Moreover, CAs are the foundation mathematical objects for other types such as sequence covering arrays (SCAs). CAs are widely accepted owing to their ability of cost-effectively executing every behavior of a system caused by the interaction of *t* or fewer input values and efficient representation of optimized *t*-way test suites. In CA, it is assumed that the array contains all required interactions among the parameters' values at least once. CA is originated from another mathematical object called orthogonal array (OA) (Federer and Mandeli 1986). All the mathematical objects reported in the literature for the representation of test suites are defined as follows.

Table 2.6 Mathematical Objects for *t*-way Test Suites and their Notations

#	**Mathematical Objects**	**Notations**
1	Orthogonal Array (OA)	$OA_\lambda(N; t, P, v)$
2	Covering Array (CA)	$CA_\lambda(N; t, P, v)$
3	Mixed Covering Array (MCA)	$MCA(N; t, P, (v_1, v_2, \dots v_P\})$ or $MCA(N; t, v^P)$
4	Variable Strength Covering Array (VCA)	$VCA(N; t; P, v, \{CA_1 \dots CA_j\})$
5	Constrained Covering Array (CCA)	$CCA(N; t, P, v)$
6	Sequence Covering Array	$SCA(N; t, P, v)$
7	Cost-Aware CA	$CTCA(N; t, P, Cost(CA))$

Definition 2.1. An $OA_\lambda(N; t, P, v)$ is an array of $N \times P$ dimensions where it is needed that every $N \times t$ sub-array contains each *t*-tuple exactly λ times where $\lambda = N/v^t$; *t* is the interaction strength, *P* is the number of parameters ($P \geq t$); and *v* is the number of values corresponding to each parameter.

For instance, $OA(9; 3, 4, 3)$ contains only nine rows in the array to cover *3*-way interactions *(t)* among a system of four parameters *(P)* each with three values *(v)*. The use of OA is discontinued in *t*-way test suite generation owing to the requirement of the exact repetition of the *t*-elements when parameters' values grow. As an alternative version of the OA, the CA notation is presented. CA can handle an increasing number of parameters and values (Kacker, Kuhn et al. 2013) and is defined as:

Definition 2.2. $CA_\lambda(N; t, P, v)$ is the general form of CA which represents an $N \times P$ array on values $(0, 1, \dots, v\text{-}1)$ | every *X* where $X = \{x_0, \dots, x_{t-1}\} \in \lambda$-covered and every

$N \times t$ sub-array consists of all ordered t size subsets over v values at least λ times, where the set of columns (parameters) $X = \{x_0, \ldots, x_{t-1}\} \supseteq \{0, \ldots, P-1\}$.

For $\lambda=1$, the notation reduces simply to $CA(N; t, P, v)$ which indicates that all the t-tuples or elements of the values of system's parameters appear in the array at least once (Hartman and Raskin 2004). A CA with fewer possible rows (i.e., with smallest N) corresponds to an optimal CA which is represented mathematically using Eq. 2.3.

$$CA(t, P, v) = \min \{ N: \exists \; CA \quad (N; t, P, v)\} \qquad 2.3$$

The notation of CA assumes that the value v (number of values) for each parameter P needs to be uniform which is not common in real applications. Usually, parameters of many real-world systems have varying number of values. Such systems can be modelled by another general mathematical structure called mixed covering array (MCA) which is defined below:

Definition 2.3. $MCA(N; t, P, (v_1, v_2, \ldots v_P))$ represents an $N \times P$ array over v values, where the rows of every $N \times t$ sub-array contain all t interactions of values at least once from the t columns.

The notation has the flexibility to be presented as $MCA(N; t, v^P)$ that can also be used for a CA with fixed-level, such as $CA(N; t, v^P)$. Both the abovementioned objects define a fixed strength t across all parameters. However, normally certain groups of parameters require much stronger testing (i.e., higher interaction strength for certain groups of parameters). This is particularly beneficial in cases, for example, when increasing t across all parameters is expensive or when developers/testers can identify that certain groups of parameters can cause failures (Yilmaz, Fouch et al. 2014). In a nutshell, mixed or variable strength covering array (VCA) offers the flexibility of varying coverage strength across the input space and is defined as:

Definition 2.4. $VCA(N; t; P, v, \{CA_1 \ldots CA_j\})$ is also a mathematical structure generated as an array of N rows and P columns on v values, but every $N \times t$ array contains one or more sub-covering arrays, namely $CA_1..CA_i$ *with* interaction strengths $t_1 \ldots t_j$, respectively, all larger than t.

All these objects are called traditional mathematical objects for *t*-way testing. Although the objects explained so far support many applications, they still may not suit the requirements of many other current applications.

The involvement of constraints in the usage of modern day software systems is now customary. Constraints dictate that certain combination of values must either be present or absent in the array. A constraint can make many of the CA or MCA rows invalid to test. To this end, another definition of constrained covering array (CCA) can be derived from the previous definitions (Ahmed, Gambardella et al. 2017):

Definition 2.5. In its standard form, *CCA(N; t, P, v)* can be defined as an $N \times P$ sub-array on *v* values with constraints *C*, where every $N \times t$ sub-array satisfies constraints of all ordered subset of size *t* over values *v* at least once.

To accommodate varying or mixed number of values, the notation *CMCA(N; t, P, (v_1, v_2, ... v_P))* is used.

To test a system with traditional mathematical objects, it is assumed that ordering of parameter values in a given row of the object has no effect on its ability of exploring fault. Any permutation of the parameter values in a row covers similar set of parameter value combinations, and need to detect similar interaction faults. However, this is not the case particularly in event-driven systems such as graphical user systems and device drivers where processing of an event is often dependent on prior events. Therefore, different permutations of the same group of events can detect different interaction faults (Ahmed and Zamli 2011b). This new mathematical object can be defined as follows (Kuhn, Higdon et al. 2012):

Definition 2.6. A sequence covering array, *SCA(N; t, P, v)* is an $N \times P$ array where each row contains *v* values from parameters *P* with every *t*-length permutation of the values *v* at least once. Adjacency of the values in the *t*-length permutations is not required.

Normally different testing cost is associated with every configuration (or test run). For instance, software installation or compilation costs more than other configurations (Yilmaz, Fouch et al. 2014). In such testing scenarios, minimizing the number of configurations or test runs may not reduce the overall cost of testing. With this, another object can be defined as follows (Demiroz and Yilmaz 2016):

Definition 7. A cost-ware covering array, *CTCA(N; t, P, Cost(CA))* is a *t*-way covering array that minimizes the cost of each row of the CA.

In this work, only CA, MCA and VCA are considered. Most of the *t*-way testing literature focused on these objects. CCA is also a very important construct as constraints are common in today's complex software systems. CTCA is a recently introduced object that attempts to minimize the number of costly interaction elements in the array.

As stated earlier, the CA notation can efficiently abstract *t*-way test suites. For example, Figure 2.6 (a) shows a test suite of size $N = 8$ (i.e., eight test cases) with *three*-way $(t = 3)$ interactions for a system that has four parameters *(P)*, namely *a*, *b*, *c*, and *d* each with two values *(v)*. This test suite can simply be represented by the notation $CA(8; 3, 2^4)$. Figure 2.6 (b) is another test suite of size $N = 12$ with *three*-way interactions for a system that has four parameters *(a, b, c, d)* where three parameters *(a, b, and c)* carry two values and one parameter *(d)* carries three values. For this test suite, the notation $MCA(12; 3, 2^3\ 3^1)$ can be used.

Similarly, Figure 2.6 (c) shows a test suite of size $N = 9$ with *two*-way (the main strength t_m) interactions for a system having four parameters *(a, b, c, and d)* of which two have three values *(a* and *b)* and two have two values *(c* and *d)*. The suite also covers *three*-way interactions (sub-strength t_s) of the system's three parameters *(b, c, and d)* where the first one *(b)* has three values, whereas the last two *(c and d)* have two values. For the representation of this complex structure, the simple notation $VCA(9; 2, 3^2\ 2^2, \{CA(3, 3^1\ 2^2)\})$ can be used.

Mathematically, the *t*-way test generation can be expressed as an optimization problem using Eq. 2.4 and Eq. 2.5:

$$\text{Maximize } f(x) = \sum_1^N x_i \qquad 2.4$$

$$\text{Subject to } x \in x_i\ , i = 1, 2, \ldots, N \qquad 2.5$$

where $f(x)$ is an objective function capturing the weight of the test case in terms of the number of covered interactions; x is the set of each decision variable x_i ; x_i is the set of possible range of values for each decision variable, that is, $x_i = \{x_i(1), x_i(2), \ldots, x_i(K)\}$ for discrete decision variables $(x_i(1) < x_i(2) < \ldots < x_i(K))$; N is the

number of decision parameters; and K is the number of possible values for the discrete variables.

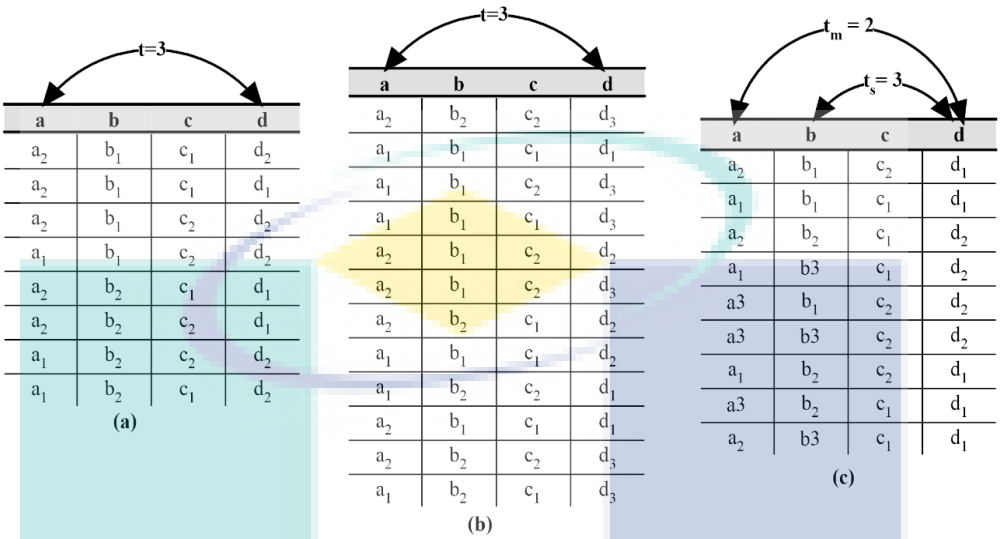


Figure 2.6 Representation of CA, MCA, and VCA

So far, this section presented the preliminaries of mixed strength and *t*-way testing that included the background and mathematical notations of the same. The next section presents state-of-the-art meta-heuristic based strategies for *t*-way test suites generation.

2.3 Meta-heuristic Algorithms

Problems are considered hard optimization problems if any deterministic algorithm fails to solve them satisfactorily within an acceptable time limit. These hard problems are everywhere, from computer science to engineering and from management to finance. There can be different categories of hard optimization problems based on whether they are single or multi-objective, discrete or continuous, static or dynamic, constrained or unconstrained. In order to solve these problems to near optimality, meta-heuristic algorithms can be used. Meta-heuristic algorithms solve approximately many hard optimization problems without the need of adapting deeply to each problem. This is clearly indicated in the name with the Greek prefix "meta" that they are "higher-level" heuristics contrary to problem-specific heuristics. Meta-heuristic algorithms are generally used for problems which have no problem-specific algorithms for their solution (Boussaïd, Lepagnot et al. 2013). They have successfully solved complex problems in many fields of study such as forecasting (Cheng, Firdausi et al. 2014, Cheng, Wibowo et al. 2015), economics (Arifovic 1996), medicine (Sheikhan and Ghoreishi 2013),

computer science (Mirjalili, Mohd Hashim et al. 2012), and engineering (Talatahari, Kheirollahi et al. 2013, Cheng, Prayogo et al. 2014), to name a few.

Characteristics common in almost all meta-heuristics algorithms include inspiration from nature (simulate laws from biology or ethology, physics or sociology); the application of stochastic constructs (involving randomness); no use of gradient or Hessian matrix of the fitness function; adaptation of several specific parameters to the optimization problem (Boussaïd, Lepagnot et al. 2013). Some of the new algorithms claim to be *parameter-free* (i.e., they do not use algorithm-specific parameters).

Different classifications for meta-heuristic algorithms are reported in the literature. Two well-known classifications are based on the inspiration of an algorithm from a natural phenomenon such as swarm intelligence, evolution theory, physics, etc., and the number of random solutions generated by an algorithm in each iteration during optimization. This last classification has two categories: single-solution based also known as trajectory algorithms and population-based meta-heuristic algorithms. Algorithms in the former category generate only one random solution and subsequently improve it stochastically till the end of optimization. Simulated Annealing (SA), the Iterated Local Search (ILS), the Tabu Search (TS), the Guided Local Search (GLS), the GRASP method, the Variable Neighborhood Search (VNS) and their variants are some meta-heuristic algorithms in this category. The focus of this work is on the latter category in which algorithms mostly generate many random solutions and improve them in each stage of optimization. Some popular algorithms in this category include Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Artificial Bee Colony (ABC), Harmony Search (HS), Cuckoo Search (CS) and Teaching Learning-based Optimization (TLBO).

Exploration and exploitation (or diversification and intensification) lie at the heart of every meta-heuristic algorithm (Talbi 2009). Exploration finds potential areas of solutions in the entire random search space, whereas exploitation focuses on the neighborhood of potential solutions. Excessive or limited use of any of the two can degrade the performance of a meta-heuristic algorithm. Therefore, a meta-heuristic algorithm needs to have a proper balance between exploration and exploitation so as to search high-quality solutions (M. Črepinšek, Liu et al. 2013, Yang, Deb et al. 2013).

2.4 Meta-heuristic-based *t*-way Strategies

Generally, meta-heuristic-based strategies start by creating lists that consist of all mixed strength and *t*-way interaction elements. The strategy employs an efficient algorithm for this purpose. A meta-heuristic algorithm such as PSO adopted by the strategy then searches an optimal test case based on the maximum coverage of interaction elements in the randomly generated population of test cases. Afterwards, the strategy removes the covered interaction elements from the list. The strategy continues this process till full coverage of all the interaction elements i.e., when the list is empty. Figure 2.7 depicts the steps common in most meta-heuristic based *t*-way test suite generation strategies.

General *t*-way test suite generation strategy based on meta-heuristic algorithm

Input: Parameters P, its values v and interaction strength t
Output: Final t-way test suite
Generate interaction elements list H_s based on P, v and t
while H_s *is not empty* **do**
    Generate a random set of test cases
    Apply the adopted meta-heuristic algorithm such as PSO, ATLBO, etc.
    Search the test case with best objective function evaluation
    Add the best test case to the final test suite
    Remove the covered interaction elements from H_s
end
Return the final test suite with minimum possible test cases

Figure 2.7 General Meta-heuristic-based Strategy for *t*-way Test Suite Generation

This section briefly reviews the mixed strength and *t*-way test suite strategies reported in the literature. As the focus of this thesis is the generation of mixed strength test suites via a meta-heuristic algorithm, the strategies based on meta-heuristic algorithms are discussed only from the related literature. Initially, *t*-way strategies based on each meta-heuristic algorithm such as PSO are reviewed. Based on this review, the test suite generation strategies are then divided into three different categories by considering the application of meta-heuristic algorithm in a strategy. Finally, the pros and cons of test data generation strategies based on each individual meta-heuristic algorithm are outlined. For an objective assessment, issues such as parameter tuning and how each meta-heuristic algorithm in a strategy balances exploration and exploitation are critically analysed.

2.4.1 Simulated Annealing-based *t*-way Strategies

Simulated Annealing (SA) is a single-solution-based meta-heuristic algorithm inspired by the physical annealing process of metals. The algorithm starts by initializing its temperature parameter and then generating an initial solution. Next, SA randomly selects a new solution in neighborhood of the current solution in each iteration. Based on the temperature and fitness function evaluations for the new and current solutions, SA either accepts or rejects the new solution. SA updates the current solution with the new one based on better fitness evaluation, otherwise, it uses probability to accept the new solution (Eaarts and Korst 1989). At the end of each iteration, SA decreases the temperature by using a cooling rate to gradually decrease the probability of accepting poor solutions. Moreover, the acceptance probability, $p\left(T, f(X^{t+1}), f(X^{t})\right) = e^{\left(\frac{f(X^{t+1}) \geq f(X^{t})}{T}\right)}$ of SA enables it to balance exploration and exploitation. The performance of SA depends on the tuning of its two control parameters, namely the temperature $T$and the cooling rate r (Busetti 2003).

As for as *t*-way testing is concerned, SA is the most widely used meta-heuristic algorithm. It has successfully generated most optimal CAs and its variants. Cohen et al. (Cohen, Colbourn et al. 2003) first employed SA for the construction of CAs. The strategy built optimal CAs as compared to algebraic methods only in the cases when interaction strength $t \leq 3$. In (Cohen, Gibbons et al. 2003a), VCAs are introduced and generated with SA-based strategy. The interaction strength is still smaller i.e., $t \leq 3$. SA is used in combination with algebraic constructions to generate the most optimal CAs, MCAs and VCAs (Cohen, Colbourn et al. 2003). Similarly, SA is hybridized with a greedy algorithm in (Bryce and Colbourn 2007) and is proved faster than a hybrid approach based on TS and greedy algorithm for the generation of CAs. Cohen et al. (Cohen, Colbourn et al. 2008) proposed a hybrid strategy based on SA and algebraic constructions to generate the smallest sizes strength $t = 3$ CAs. However, the strategy failed when no algebraic construction was possible for an array. The SA search is reformulated in (Garvin, Cohen et al. 2009) so that it can work better on both constrained and unconstrained problems. However, even offering better results (25% fewer configurations on average), run time of the SA is longer. The binary search of the strategy suffered from a faulty assumption and was less efficient. Binary CAs of strength $t \leq 5$ are generated by SA based strategy integrated with binary composite functions in (Torres-

Jimenez and Rodriguez-Tello 2010). The strategy generated the most optimal binary CAs in many cases.

CASA (Covering Arrays by Simulated Annealing) is the most successful SA-based constrained test suite generation strategy. The eight modifications (where modifying the global strategy for selecting the sample size and changing the neighboring of the search being the most promising) enabled CASA to be more efficient. The strategy outperformed, with these modifications, even greedy based strategy in terms of performance in certain cases. Another SA-based strategy called Improved Simulated Annealing (ISA) proposed in (Rodriguez-Cristerna and Torres-Jimenez 2012) successfully generated binary CAs of strength $3 \leq t \leq 6$. The number of parameters that ISA can support is between 4 and 1712. To generate MCAs of strength $2 \leq t \leq 3$, Avila-George et al. (Avila-George, Torres-Jimenez et al. 2013) proposed SA based strategy that showed competitive results as far as performance is concerned. The strategy used the Diophantine equation to fine tune its control parameters.

SA-VNS (Rodriguez-Cristerna and Torres-Jimenez 2012) generated quality MCAs of strength $2 \leq t \leq 3$ and is based on two meta-heuristic algorithm: SA and Variable Neighborhood Search (VNS). The generation time of SA-VNS was longer than those strategies selected from references. In another similar research (Rodriguez-Cristerna, Torres-Jimenez et al. 2015), SA and VNS are again collectively used for the generation of MCAs. In this hybrid implementation, SA is responsible to control acceptance moves, whereas VNS is mainly responsible to avoid local optimal arrays by searching neighbourhoods at various distances. The strategy required to fine tune seven different parameters. SA is used as a hyper-heuristic for the selection of six search operators as low-level heuristics to generate CCAs by Jia et al. (Jia, Cohen et al. 2015). This is the first hyper-heuristic methodology proposed for the generation of combinatorial test suites. Apart from showing effectiveness in terms of both efficiency and performance, the hyper-heuristic based strategy is general as it learns the nature of the problem by selecting the appropriate search operator. More recently, the SA-based strategy (Demiroz and Yilmaz 2016) is used to generate cost-aware covering arrays CTCAs that not only minimizes the number of test cases but also the associated interaction test cost. The empirical evidence suggested that the proposed strategy

outperformed existing strategies. SA based strategy for the test suite generation is shown in Figure 2.8.

SA for test suite generation

Input: An array of $N \times k$ rows
Output: Covering array

```
1  let A = initialState(t, k, v, C, N);
2  let temperature = initialTemperature;
3  while termination criteria not satisfied do
4  |  choose row from 1...N;
5  |  choose column from 1...k;
6  |  choose symbol from v_column;
7  |  A' = A;
8  |  A'_{row,column} = symbol;
9  |  Δfitness = countNoncoverage(A') − countNoncoverage(A);
10 |  if Δfitness ≤ 0 then
11 |  |  A = A';
12 |  end
13 |  else
14 |  |  A = A' with probability e^{Δfitness/temperature};
15 |  end
16 |  temperature = cool(temperature);
17 end
18 return A
```

Figure 2.8 SA-based Strategy for *t*-way Test Suite Generation
Source: Garvin, Cohen et al. (2011)

SA has been reported as one of the most effective meta-heuristic algorithms for the generation of CAs and MCAs. Its application for test suit generation in stand-alone or hybrid form will further be explored by the research community (Timaná-Peña, Cobos-Lozada et al. 2016). However, SA is not only a single-solution based meta-heuristic but also rely hugely on the neighborhood structures while such structures are not available (Beasley, Martin et al. 1993).

2.4.2 Tabu Search-based *t*-way Strategies

Tabu search (TS) (Glover 1989) is another single-solution meta-heuristic algorithm inspired by the human memory. TS conducts the searching process by using neighborhood structures equivalent to short-term and long-term human memories. Different neighborhood structures are used to remember the path visited by the algorithm during the search. A *tabu list* is maintained by TS to remember recently encountered solutions and subsequently forbid their regeneration. The list acts as a short-term memory of the algorithm, prevents endless repetition and forces TS to accept worse moves. To

balance exploitation and exploration, the algorithm can introduce short-term memory and long-term memory.

TS is adopted by many test suite generation strategies. Nurmela (Nurmela 2004) proposed a TS-based strategy that improved many available results i.e., sizes of CAs at the expense of longer run time. In (Bryce and Colbourn 2007), TS is combined with a greedy algorithm for *t*-way test suite generation. However, this hybrid strategy was slower in covering *t*-tuples than the SA-based hybrid strategy. The covering perfect hash families (CPHF) based representation of CAs enable the TS-based strategy by (Walker Ii and Colbourn 2009) to generate optimal arrays of higher strengths ($t \geq 5$). Gonzalez-Hernandez et al. (Gonzalez-Hernandez, Rangel-Valdez et al. 2010) proposed first TS-based strategy that can generate MCAs with more parameters (from 2 to 11), values (from 2 to 20) and *t* (from 2 to 6). However, the strategy suffers from extensive tuning of configuration probabilities for TS to achieve optimal results. MiTS (Gonzalez-Hernandez and Torres-Jimenez 2010) is a TS-based strategy applied to a limited set of MCAs (10 only) of strength $2 \leq t \leq 3$. In addition, it requires tuning of several components of its three neighborhood functions.

Tabu Search Algorithm (TSA) (Gonzalez-Hernandez, Rangel-Valdez et al. 2012) is a TS-based strategy with four neighborhood functions. The strategy successfully generated most optimal sizes 20 MCAs of strength $2 \leq t \leq 6$ out of total 23 MCAs used as benchmarks. TSA requires rigorous two steps fine tuning process to find the best configurations for its three main components. TCA (Lin, Luo et al. 2015) used a two-mode local search framework that combines TS with the random walk to generate CCAs. The strategy outperformed its meta-heuristic based and greedy based competitors on *3*-way and *2*-way CCAs. Smaller sizes MCAs of strength $2 \leq t \leq 3$ are generated by a TS-based strategy (Gonzalez-Hernadez 2015). The novelty of the proposed strategy is its application of statistical tests to fine tune related parameters. The hyper-heuristic strategy (Zamli, Alkazemi et al. 2016) adopted TS as its high-level heuristic. As a selection and acceptance hyper-heuristic, the strategy generated CAs and MCAs of strength $2 \leq t \leq 6$. This TS-based hyper-heuristic strategy outperformed many strategies based on meta-heuristic algorithms and other tools on a wide range of benchmarks as far as efficiency is concerned. Statistical evidence also showed the effectiveness of the strategy. The pseudo code of a TS-based strategy (MiTS) is shown in Figure 2.9.

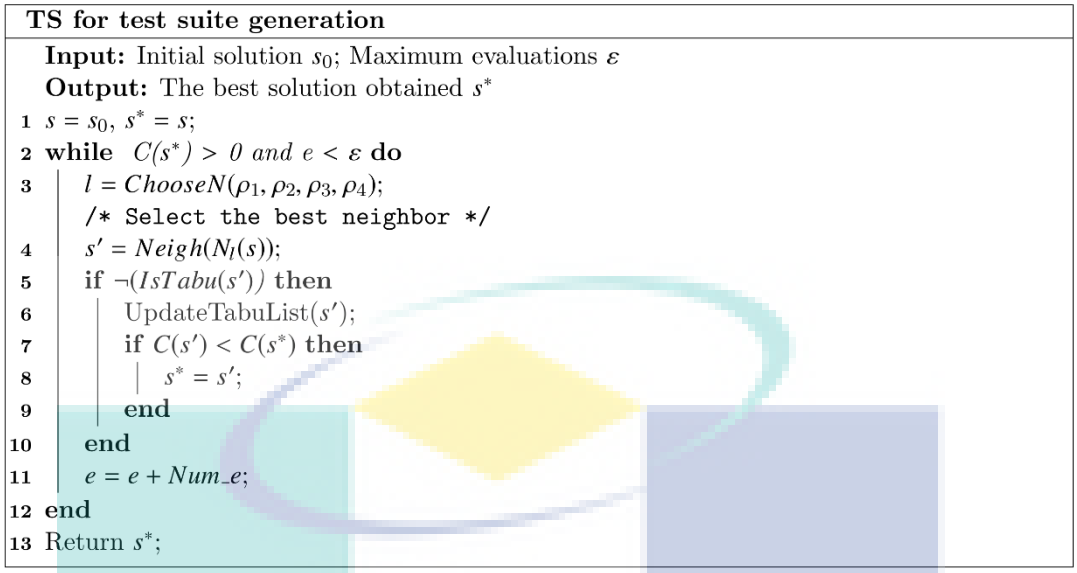


Figure 2.9 TS-based Strategy (MiTS) for *t*-way Test Suite Generation

Source: Gonzalez-Hernadez (2015)

2.4.3 Genetic Algorithm-based *t*-way Strategies

Genetic algorithm (GA) (Holland 1975) is a well-known population-based meta-heuristic algorithm inspired by the process of natural selection. The population of solutions generated by GA evolves through generations (i.e., iterations). Best solutions survive for next generations based on objective function evaluations. Future generations are created by the successive applications of genetic operators such as crossover, mutation, and selection. During evolution, GA preserves the diverse solutions of the population to ensure adequate exploration of the search space and to escape local optimal (Boussaïd, Lepagnot et al. 2013). To balance exploration and exploitation, however, GA requires proper tuning of mutation rate, crossover rate, number of generations, and population size.

Population size is one of the main issues in GA that needs proper attention. GA with small population size could easily trap in local optima, whereas require more computation time in the case of large population size. It is observed that a population of 100 chromosomes or candidate solutions for 200 iterations or generations is an optimal population size. Increasing the population size, for example to over 1000, could not make any difference in the results (Roeva, Fidanova et al. 2013). Other research findings regarding the population size suggest that a population of 30 chromosomes are sufficient for most optimization problems (Grefenstette 1986).

In the GA-based strategy (Shiba, Tsuchiya et al. 2004), a chromosome is similar to a test case. The strategy starts by initializing a random population of m test cases. These test cases then undergo selection, crossover and mutation operations repeatedly till satisfaction of termination criteria. The algorithm of the strategy selects σ number of elite chromosomes based on the fitness evaluation for the next generations. The remaining m-σ chromosomes are used to generate the next population. Tournament selection is used for selecting two chromosomes randomly and then adding the winner to the mating pool for reproduction. The crossover enables two chromosomes to generate a new chromosome by independently exchanging values between them with probability 0.5. The mutation process replaces one position value with another value selected at random. Finally, the strategy performs a massive mutation at the end of a specified number of generations when there is no improvement. The pseudo code of GA-based strategy is presented in Figure 2.10.

GA for test suite generation

Input: A test set
Output: A test case
1 Create the initial population of P of candidates
2 Evaluate P
3 **while** *stopping criteria not satisfied* **do**
4 | Select Elite consisting of σ best individuals from P
5 | Apply *Selection* to individuals in P to create P_{mating}, consisting of $(m - \sigma)$ individuals
6 | *Crossover* P_{mating}
7 | *Mutate* P_{mating}
8 | Copy all the individuals of P_{mating} to P, replacing the worst $(m - \sigma)$ individuals in P
9 | Evaluate P
10 | **if** *stagnation criteria satisfied* **then**
11 | | Mutate P massively
12 | **end**
13 **end**
14 Return the best test case found

Figure 2.10 GA-based Strategy for *t*-way Test Suite Generation
Source: Shiba, Tsuchiya et al. (2004)

Few strategies adopted GA for *t*-way test suite generation. The work of (Shiba, Tsuchiya et al. 2004) based on GA provided competitive results but the optimality of the obtained results is not always true. To generate binary CAs of strength 3, a GA-based (i.e., memetic algorithm) strategy is introduced in (Rodriguez-Tello and Torres-Jimenez 2009). This strategy outperformed the referenced strategies by generating the most optimal CAs than known previously. The GA based strategy of (Sabharwal, Bansal et al. 2016) is an extension of the open source pairwise test generation tool called PWiseGen.

A modified binary search operator is used to overcome the restriction of using N (test size) as input to the tool in advance. Owing to the complex crossover and mutation operations, the proposed strategy takes a long time to generate higher strength CAs and MCAs which is, however, compensated by the smaller sizes of the obtained arrays. More recently, Genetic Strategy (GS) (Esfandyari and Rafe 2018) adopted GA as its backbone algorithm for generation of CAs, MCAs and VCAs. GS supports highest strength (i.e., t = 20). Based on the extensive experimental results, GS showed competitive performance (time) and efficiency (generated test suites sizes) against its competitors. Moreover, the strategy minimized the running time of the objective function evaluation by introducing an efficient bit structure.

2.4.4 Ant Colony Algorithm-based *t*-way Strategies

Ant colony algorithm (ACA) is a swarm-based meta-heuristic algorithm inspired by the foraging behavior of ants. A given optimization problem in ACA is encoded as a construction graph. The paths between the food source and ants' nest represent candidate solutions in ACA. The deposited amount of pheromone by an ant on each vertex or edge of the path determines the quality of a candidate solution. At a given point (vertex), the probability to choose an edge out of several edges is highest for the edge with the greatest concentration of pheromone.

Edge selection and pheromone update are two key operations in ACA. The first operation is based on probability, whereas the second operation is based on the movements of ants from one node to the next. For optimality, several control parameters of ACA such as pheromone amount (τ), pheromone coefficient (α), heuristic coefficient (β), and pheromone evaporation rate (ρ) need to be tuned properly.

For test suite generation, ACA represents values by food sources, whereas their locations represent parameters (Shiba, Tsuchiya et al. 2004). Test cases are represented by paths to the food sources. The amount of pheromones deposited by ants on each path determines its quality. The density of pheromones gets higher for some paths as more and more ants choose these paths over time. After comparing objective functions of these paths, the comparison algorithm of the strategy selects best test cases to be part of the final test suite. Figure 2.11 summarizes the ACA-based test suite generation strategy.

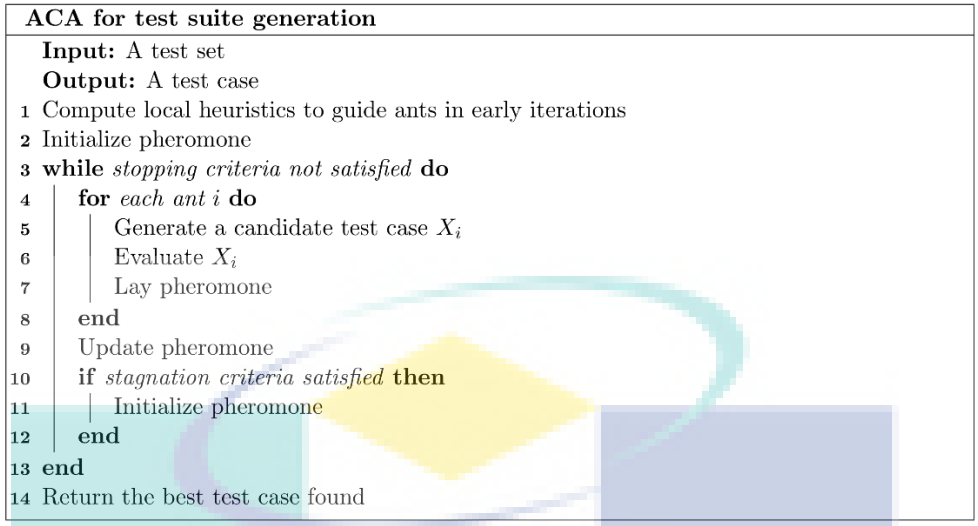


Figure 2.11 ACA-based Strategy for *t*-way Test Suite Generation

Source: Shiba, Tsuchiya et al. (2004)

The strategy proposed by (Shiba, Tsuchiya et al. 2004) first adopted ACA for the generation of CAs and MCAs of strength $2 \leq t \leq 3$. The proposed ACA based strategy showed competitive performance against GA-based strategy and other referenced strategies except the SA-based strategy. The strategy (Chen, Gu et al. 2009) adopted ACA for the generation of VCAs. Compared to the referenced SA-based and greedy-based strategies and tools, the proposed strategy showed acceptable performance.

2.4.5 Particle Swarm Optimization-based *t*-way Strategies

Particle swarm optimization (PSO) is a swarm-based meta-heuristic algorithm inspired by the flocking behavior of birds or fishes. PSO stochastically generate many candidate solutions in the search space known as particles. Each particle has a velocity, a position in the search space, and memory for remembering its last best position. PSO uses two topologies known as *gBest* (i.e., global best) and *lBest* (i.e., local best). The *gBest* evaluate a target particle against the best particle in the entire population. The *lBest*, on the other hand, checks the optimality of a target particle against its neighbouring particles. To balance exploration and exploitation, inertia weight ω is introduced in PSO. A large value of ω encourages more global search, whereas a smaller ω encourages more local search. For optimal results, PSO needs tuning of its five different parameters including cognitive parameters (*C1* and *C2*), *ω*, population size and number of iterations (Ahmed and Zamli 2011c). PSO has widespread applications in many fields of research.

Initially, Ahmed et al. adopted PSO for *t*-way test suite generation by proposing Particle Swarm-based Test Generator (PSTG) (Ahmed and Zamli 2010) and Variable Strength PSTG (VS-PSTG). PSTG successfully generated CAs and MCAs, whereas VS-PSTG generated VCAs. Both the strategies supported interaction strength up to $2 \leq t \leq 6$. The strategies, however, required extensive tuning of the related parameters to obtain optimal results. Similarly, both the strategies suffered from the problem of falling in local minima.

A set based discrete PSO (DPSO) and conventional PSO (CPSO) (Wu, Nie et al. 2015) generated a wide range of CAs, MCAs and VCAs. The systematic guidelines for tuning the parameters of both DPSO and CPSO enable the strategies to obtain optimal results. CPSO showed better performance than DPSO, whereas in efficiency DPSO produced quality *t*-way test suites. DPSO suffered from the overhead of the two auxiliary methods (re-initialization of the particles and an extended evaluation of the *gBest*). More recently, the multi-objective PSO-based strategy proposed by (Ahmed, Gambardella et al. 2017) generated CCAs of comparable sizes. For efficient performance, the strategy used multi-threading to operate all the algorithms in parallel.

Concerning *t*-way test suite generation, test cases are represented as particles in PSO-based strategies (Ahmed, Zamli et al. 2012, Ahmed, Zamli et al. 2012). A PSO based strategy starts by randomly generating a population of particles with random initial velocities. At each cycle of the search process, PSO uses the best test case to update the velocities of the particles as they fly around the search space. Based on these updated velocities, current test cases move to new test cases. These movements are continued until all interactions are covered (i.e., termination criteria for the strategy). Finally, the strategy adds the best test case in the population to the final test suite and subsequently removes the number of interactions covered by it from the list of all interactions. The pseudocode of PSO for generating test suites is shown in Figure 2.12.

One of the main disadvantages of PSO is its frequent interaction with the environment as it continuously updates the particles' velocities in the swarm until the search for a quality solution is successful. Similarly, PSO requires proper tuning of its various parameters before obtaining optimal solutions of a given optimization problem (Pedersen 2010). Moreover, the time cost of PSO is higher for practical usage (Lee and Park 2006).

PSO for test suite generation

Input: Parameters P, values v and interaction strength t
Output: Final t-way test suite

Generate all combinations list P_s
Let T_s be a set of candidate tests
while *all interactions not covered* **do**
    Randomly initialize particles X_i^t and velocities V_i^t
    for *certain iterations* **do**
        Evaluate $f(X_i^t)$ for maximum interaction coverage
        if $f(X_i^t)$ *is better* **then**
            Add X_i^t to final test suite
            Remove covered interactions from P_s
            continue
        end
        else
            choose best coverage particle to be $lBest$
            Calculate V_i^{t+1} according to $lBest$
            Move X_i^t to X_i^{t+1} according to V_i^{t+1}
        end
        Evaluate $f(X_i^{t+1})$
        if $f(lBest^{t+1})$ *is better* **then**
            $lBest = lBest^{t+1}$
        end
    end
    Let $gBest$ be the best test case found
    $gBest = lBest^{t+1}$
    Add $gBest$ to the test set T_s
    Remove covered interactions from P_s
end
return the final t-way test suite

Figure 2.12 PSO-based Strategy for *t*-way Test Suite Generation

Source: Ahmed, Zamli et al. (2012)

2.4.6 Harmony Search-based *t*-way Strategies

Harmony search (HS) is a population-based meta-heuristic algorithm inspired by the improvisation procedure of professional jazz musicians (Geem and Kim 2001). Solutions in HS are represented by harmonics. HS uses three main parameters to search perfect harmonics i.e., global optimal solutions. The harmony memory accepting rate corresponding to the usage of the harmony memory parameter in HS ensures the selection of best harmonics for the new harmony memory. Pitch adjustment enables HS to generate a new solution by slightly modifying the current solution. Pitch adjustment rate and pitch bandwidth control this parameter in HS. The randomization parameter of HS increases the diversity of solutions. HS provides balance between exploration and exploitation with the help of these parameters.

HS, like PSO, interacts frequently with the environment during the search process. It probabilistically uses values of harmony memory accepting rate r_{accept} and Pitch adjustment rate r_{pa} in order to select solutions from the Harmony Memory (HM). Some disadvantages of HS include its minimal application of mathematics and use of some kind of elitism and/or selection like GA (Yang 2010a). Moreover, the performance of HS is dependent on proper tuning of its four algorithm specific parameters, namely iterations/improvisations, harmony memory size (HMS), r_{accept} , and r_{pa}.

Concerning t-way test suite generation, (Alsewari and Zamli 2012) adopted HS in their proposed strategy called Harmony Search-based Strategy (HSS). HSS generated CAs, MCAs and VCAs with constraints support and interaction strength support of up to $2 \leq t \leq 15$, the highest after the recently introduced GA based strategy (Esfandyari and Rafe 2018). However, HSS required extensive experimentation to fine tune its various algorithm-specific parameters before generating optimal arrays. The pseudo code of HSS is presented in Figure 2.13.

2.4.7 Cuckoo Search-based *t*-way Strategies

Cuckoo search (CS) (Yang and Deb 2009) is a relatively new population-based meta-heuristic algorithm inspired by the aggressive reproduction method of some cuckoo birds. Solutions in CS are represented by cuckoos' eggs or nests. To balance between exploration and exploitation, CS intensifies the search through the use of local random walk for solutions near potential optimal solutions, whereas it employs global random walk by using Lévy flights to efficiently explore the entire search space. A switching parameter p_a controls both these walks which is the only parameter of CS that requires proper tuning.

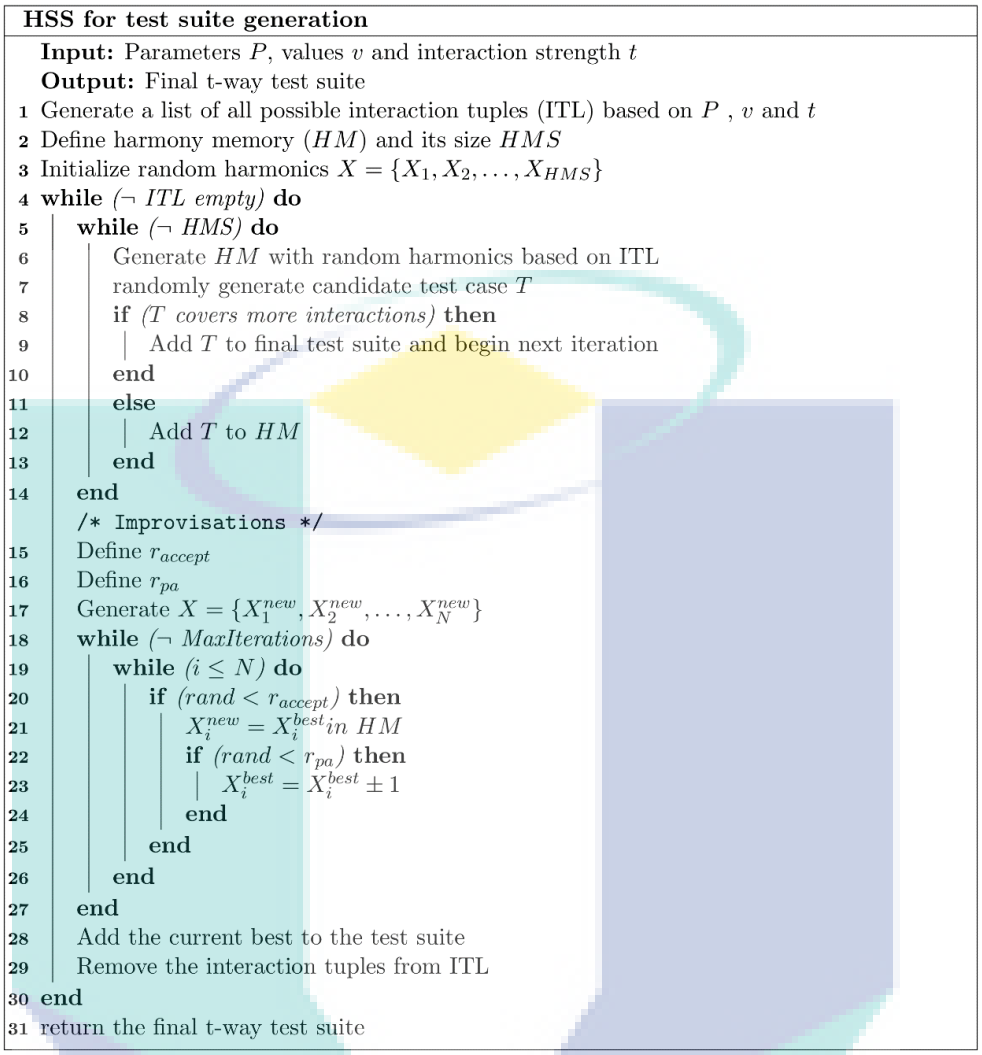


Figure 2.13 HS-based Strategy (HSS) for t-way Test Suite Generation

Source: Alsewari and Zamli (2012)

Most recently, (Ahmed, Abdulsamad et al. 2015, Ahmed 2016) adopted CS for the generation of CAs and MCAs of strength $2 \leq t \leq 6$. Both the CS-based strategies showed comparable performance and efficiency against the referenced tools and other meta-heuristic-based strategies. At the beginning, the CS-based strategy randomly generates an initial population of nests. Here, each nest represents a candidate test case. In each cycle during the search, CS first generates a new nest by employing a Lévy flight and then replaces it with the current nest based on better objective function evaluation. Afterwards, CS identifies and subsequently removes the worse nests with probability p_a. Similar to GA, remembering and considering previous best solutions (i.e., the elitism mechanism) can also be noticed in CS. Figure 2.14 shows a simple framework of CS

based strategy proposed by Ahmed et al. for the problem of generating uniform *t*-way test suites.

CS for test suite generation

Input: Parameters P, values v and interaction strength t
Output: A t-way test suite

```
Generate a list of all possible interaction tuples (ITL) based on P, v and t
Initialize the m random nests X = {X_1, X_2, ..., X_m}
while (¬ ITL empty) do
    for (¬ MaxIteration) do
        randomly select X_i^t
        Generate new egg X_i^{t+1} from X_i^t based on lévy flight
        if f(X_i^{t+1}) is better than f(X_i^t) then
            X_i^t = X_i^{t+1}
        end
        Identify and remove (based on p_a) poor nests
        Rank the best nests and find the current best
    end
    Add the current best to the test suite
    Remove the interaction tuples from ITL
end
return the t-way test suite
```

Figure 2.14 CS-based Strategy for *t*-way Test Suite Generation
Source: Ahmed, Abdulsamad et al. (2015)

2.4.8 Bat Algorithm-based *t*-way Strategies

Bat algorithm (BA) is a population-based meta-heuristic algorithm inspired by the echolocation qualities of microbats (Yang 2010b). It is one of the most simple, easy to implement, and flexible optimization algorithms that guarantees global convergence under the appropriate settings of its parameters (Huang, Zhao et al. 2013). The use of frequency-tuning enables BA to increase solutions diversity in the search space, whereas variations in loudness and pulse emission rate enable it to intensify the search into the regions with potential solutions. To be specific, BA balances between exploration and exploitation by using these parameters. Some capabilities of BA are, however, similar to SA such as the use of a constant in loudness α is similar to the cooling factor used by SA in a cooling schedule. Moreover, BA uses frequencies and locations for updating solutions like PSO.

The test suite generation strategy called BTS (Alsariera and Zamli 2015) adopted BA as its backbone meta-heuristic algorithm. BTS generated small CAs with interaction strength support of $2 \leq t \leq 6$. It has shown competitive performance against the publicly available *t*-way test suite generation tools. Test cases in BTS are represented as bats. The

strategy starts by initializing the BA parameters (i.e., frequency Q_i, loudness A_i, and pulse rate r_i) and randomly generating the population of bats with initial velocities v_i. At each iteration, BTS selects the best test case based on the maximum coverage of interaction tuples and updates the frequencies, locations, and velocities of others in the population. Upon searching a best test case, the strategy removes the interaction tuples covered by it from the interaction elements list *(IEL)*. Figure 2.15 presents BTS for the generation of CAs.

BA for t-way test suite generation

Input: Covering array notation
Output: Final t-way test suite ($FTLS$)

```
Initialize BA attributes
Generate interaction element list (IEL)
Initialize BA population randomly, where bats are test cases (tc)
while IEL is not empty do
    while T < nGeneration do
        Select tc as best test case with best objective function from the population
        for all nBats do
            Calculate frequency Q_i
            Get the value v_{i+1} according to old v_i, x_i and Q_i
            Move x_i to x_{i+1} according to new v_{i+1}
            if best x_{i+1} covers maximum interaction elements then
                Accept new tc as best x_i = best x_{i+1}
                Increase r_i and reduce A_i
            end
            Accept best x_{i+1} tc as the best test case
        end
        Evaluate objective function for all test cases
    end
    Add best tc to the FTSL
    Remove the covered tc from IEL
end
Return the final test suite
```

Figure 2.15 BA-based Strategy (BTS) for *t*-way Test Suite Generation

Source: Alsariera and Zamli (2015)

2.4.9 Bees Algorithm-based *t*-way Strategies

Bees Algorithm (Pham, Ghanbarzadeh et al. 2006) is a population-based meta-heuristic algorithm inspired by the foraging behavior of honey bees. To balance exploration and exploitation, the Bees Algorithm divides the search space into patches based on the objective function evaluations. The algorithm intensifies search by recruiting more bees for patches with better objective function values. Generally, the Bees Algorithm favors local search more than global search. For optimal results, six different parameters of the algorithm need to be tuned which are the number of scout bees

(n), the number of patches *(m)*, the number of elite patches *(e)* out of *m*, the number of non-elite patches *(nsp)*, the number of bees recruited for elite patches *(nep)*, and an initial size of patches *(ngh)*.

The strategy proposed by (Mohd Hazli, Zamli et al. 2012) adopted Bees Algorithm to generate optimal SCAs. Test cases are represented by patches. The strategy showed competitive results against the only available SCAs generation framework for a small number of systems. Bees Algorithm based strategy (Mohd Hazli and Zamli 2013) successfully generated CAs of strength $3 \leq t \leq 10$. Figure 2.16 presents the pseudo code of the strategy based on Bees Algorithm.

Bees Algorithm for t-way test suite generation

Input: Number of events e, interaction strength t
Output: Final Sequence t-way test suite (FTS)

```
Generate interaction elements list (IE)
while IE is not empty do
    Initialize bees population N randomly
    while ¬MaxLoop do
        Identify m best solutions for neighborhood search based on the objective function
        for each elite patch e in m do
            Send nep bees for each patch
            Evaluate objective functions of the bees
        end
        for each non-elite patch in m − e do
            Send nsp bees for each patch
            Evaluate objective functions of the bees
        end
        Update population N accordingly
    end
    Select the fittest bee and add it to the final test suite
    Remove covered tuples from IE
end
```

Figure 2.16 Bees Algorithm-based Strategy for *t*-way Test Suite Generation
Source: Mohd Hazli, Zamli et al. (2012)

2.5 Categories of Meta-heuristic-based *t*-way Strategies

Based on the review in the previous section, meta-heuristic-based test suite generation strategies can be classified into three categories: standard strategies, hybrid strategies, and adaptive strategies.

A strategy is categorized as a standard strategy if it generates *t*-way test suites by employing only a single meta-heuristic algorithm. For instance, Ahmed et al. (Ahmed, Zamli et al. 2012) and Wu et al. (Wu, Nie et al. 2015) adopted only PSO for generating

test suites. Moreover, Cohen et al. (Cohen, Gibbons et al. 2003a) and Alsewari et al. (Alsewari and Zamli 2012) employed SA and HS, respectively, to generate VCAs and CCAs. These and others standard meta-heuristic-based strategies have successfully generated CAs and its variants of competitive sizes. Some of the most widely cited earlier standard meta-heuristic-based strategies in *t*-way test suite generation are Tabu search (TS) (Nurmela 2004), SA (Cohen, Gibbons et al. 2003a) and so forth. More recently, PSO (Ahmed and Zamli 2011c, Wu, Nie et al. 2015, Ahmed, Gambardella et al. 2017), HS (Alsewari and Zamli 2012), CS (Ahmed, Abdulsamad et al. 2015) and BA (Alsariera and Zamli 2015) have been adopted by *t*-way strategies. Table 2.7 summarizes these strategies reported in the literature between 2003 and 2018.

Table 2.7 Standard Meta-heuristic-based Strategies

#	Reference	Adopted meta-heuristic algorithm	Generated CAs and its variants	Interaction strength support (*t*)
1	(Cohen, Gibbons et al. 2003b)	SA	CAs only	$2 \leq t \leq 3$
2	(Cohen, Gibbons et al. 2003a)	SA	CAs, MCAs, VCAs	$2 \leq t \leq 3$
3	(Nurmela 2004)	TS	CAs only	$2 \leq t \leq 3$
4	(Rodriguez-Tello and Torres-Jimenez 2009)	GA	Binary CAs only	$2 \leq t \leq 3$
5	(Garvin, Cohen et al. 2009)	SA	CAs, MCAs, CCAs	$2 \leq t \leq 3$
6	(Ahmed and Zamli 2010)	PSO	CAs, MCAs	$2 \leq t \leq 6$
7	(Ahmed and Zamli 2011c)	PSO	CAs, MCAs, VCAs	$2 \leq t \leq 6$
8	(Mohd Hazli, Zamli et al. 2012)	Bees Algorithm	SCAs only	$2 \leq t \leq 10$
9	(Ahmed, Zamli et al. 2012)	PSO	CAs, MCAs, VCAs	$2 \leq t \leq 6$
10	(Alsewari and Zamli 2012)	HS	CAs, MCAs, VCAs, CCAs	$2 \leq t \leq 15$
11	(Mohd Hazli and Zamli 2013)	Bees Algorithm	CAs	$4 \leq t \leq 10$
12	(Wu, Nie et al. 2015)	PSO	CAs, MCAs VCAs	$2 \leq t \leq 4$
13	(Ahmed, Abdulsamad et al. 2015)	CS	CAs, MCAs	$2 \leq t \leq 4$
14	(Alsariera and Zamli 2015)	BA	CAs	$2 \leq t \leq 6$
15	(Sabharwal, Bansal et al. 2016)	GA	CAs, MCAs	$2 \leq t \leq 4$
16	(Ahmed 2016)	CS	CAs, MCAs	$2 \leq t \leq 6$
17	(Demiroz and Yilmaz 2016)	SA	CTCAs	$2 \leq t \leq 3$
18	(Ahmed, Gambardella et al. 2017)	Multi-objective PSO	CCAs	$2 \leq t \leq 3$
19	(Esfandyari and Rafe 2018)	GA	CAs, MCAs, VCAs	$2 \leq t \leq 20$

Hybrid meta-heuristic-based strategies is the second category that combines a meta-heuristic with another algorithm (meta-heuristic or otherwise) to further increase the efficiency of *t*-way strategies. Table 2.8 presents these *t*-way strategies.

Table 2.8 Hybrid Meta-heuristic-based Strategies

#	Reference	Adopted meta-heuristic algorithm (s) and/or other methods	Generated CAs and its variants	Interaction strength support (*t*)
1	(Cohen, Colbourn et al. 2003)	SA, algebraic constructions	CAs, MCAs	$2 \leq t \leq 3$
2	(Shiba, Tsuchiya et al. 2004)	GA, ACA, test case minimization algorithm	CAs, MCAs,	$2 \leq t \leq 3$
3	(Bryce and Colbourn 2007)	SA, TS, HC, Great Flood, greedy algorithm	CAs, MCAs	$2 \leq t \leq 4$
4	(Cohen, Dwyer et al. 2007)	SA, greedy algorithm, SAT solver	CAs, CCAs	$2 \leq t \leq 3$
5	(Walker Ii and Colbourn 2009)	TS, Covering Perfect Hash Families (CPHF)	CAs	$2 \leq t \leq 7$
6	(Chen, Gu et al. 2009)	ACA, tests minimization algorithm	CAs, MCAs, VCAs	$2 \leq t \leq 3$
7	(Gonzalez-Hernandez, Rangel-Valdez et al. 2010)	TS, two neighbourhood functions	CAs, MCAs	$2 \leq t \leq 6$
8	(Gonzalez-Hernandez and Torres-Jimenez 2010)	TS, three neighbourhood functions	CAs, MCAs	$2 \leq t \leq 3$
9	(Torres-Jimenez and Rodriguez-Tello 2010)	SA, composite neighbourhood functions	Binary CAs	$2 \leq t \leq 5$
10	(Garvin, Coehn et al. 2011)	SA, efficient one-sided narrowing algorithm	CAs, MCAs, CCAs	$2 \leq t \leq 3$
11	(Rodriguez-Cristerna and Torres-Jimenez 2012)	SA, Variable Neighbourhood Search (VNS)	CAs, MCAs	$2 \leq t \leq 3$
12	(Rodriguez-Cristerna and Torres-Jimenez 2012)	SA, neighbourhood functions	Binary CAs	$2 \leq t \leq 3$
13	(Rodriguez-Cristerna, Torres-Jimenez et al. 2015)	SA, variable neighbourhood search (VNS)	CAs, MCAs	$2 \leq t \leq 3$
14	(Jia, Cohen et al. 2015)	SA, six search operators as low level heuristics	CCAs	$2 \leq t \leq 3$
15	(Lin, Luo et al. 2015)	TS, random walk	CCAs	$2 \leq t \leq 3$
16	(Zamli, Alkazemi et al. 2016)	TS, four low-level meta-heuristic search operators	CAs, MCAs	$2 \leq t \leq 6$
17	(Avila-George, Torres-Jimenez et al. 2018)	SA, algebraic method	CAs	$2 \leq t \leq 6$

For instance, augmented annealing (Cohen, Colbourn et al. 2008) combined SA with algebraic construction to find much smaller arrays faster. The hybrid *t*-way strategy (Bryce and Colbourn 2007) initially employed a greedy algorithm and then TS to generate CAs. Similarly, two strategies by (Rodriguez-Cristerna and Torres-Jimenez 2012,

Rodriguez-Cristerna, Torres-Jimenez et al. 2015) hybridized SA with VNS to generate MCAs and CAs. Finally, (Jia, Cohen et al. 2015, Zamli, Alkazemi et al. 2016) explored hybridization of meta-heuristic algorithms based on hyper-heuristic methodology. These strategies employed SA and TS, respectively as their high-level heuristics to select an appropriate low-level meta-heuristic algorithm or search operator from a pool of available meta-heuristic algorithms or search operators.

The third category is named adaptive meta-heuristic algorithms-based strategies. Strategies in this category employ meta-heuristic algorithm(s) for test suite generation with an additional method to further improve the performance of the employed meta-heuristic algorithm by dynamic tuning of its control parameters.

A strategy proposed by Mahmoud and Ahmed (Mahmoud and Ahmed 2015) is one example in this category. To generate smaller CAs and MCAs, this strategy adopted fuzzy inference system to automate the parameter tuning of PSO. MiTS (Gonzalez-Hernadez 2015) generates smaller MCAs of uniform strength, is also an example of adaptive meta-heuristic-based *t*-way strategy. To achieve optimal results, the tuning procedure of the strategy uses statistical tests to identify best values for the search parameters. Table 2.9 lists these adaptive *t*-way strategies.

Table 2.9 Adaptive Meta-heuristic-based Strategies

#	Reference	Adopted meta-heuristic algorithm and other method	Generated CAs and its variants	Interaction strength support (*t*)
1	(Avila-George, Torres-Jimenez et al. 2013)	SA, neighbourhood two functions, Diophantine equation for tuning control parameters	CAs, MCAs	$2 \leq t \leq 3$
2	(Gonzalez-Hernadez 2015)	TS, neighbourhood functions, statistical method for parameter tuning	CAs, MCAs	$2 \leq t \leq 6$
3	(Mahmoud and Ahmed 2015)	PSO, Mamdani fuzzy inference system	CAs, MCAs	$2 \leq t \leq 4$

The motivation behind standard meta-heuristic-based strategies is to avoid the restrictions or requirement of priori knowledge by algebraic methods for generating CAs and its variants. In practical testing scenarios, fulfilling all the CAs generation

requirements is often not feasible (Wu, Nie et al. 2015). Parameter tuning or the application of appropriate search operator is a common challenge for the majority of meta-heuristic algorithms. In the case of *t*-way testing, customization of the algorithmic parameters of the adopted meta-heuristic algorithms is required prior to obtaining optimal test suite. This is especially essential when the nature of the problem changes. For instance, different search operators need to be applied when constraints are involved in the data set (Kitsos, Simos et al. 2015).

Standard meta-heuristic-based *t*-way strategies have successfully created optimal arrays in many cases. Unfortunately, the creation time of these optimal arrays is considerably high. The motivation behind hybrid strategies is to reduce this time and enhance the efficiencies of the strategies. The challenge for this category is its confinement to only smaller interaction strengths i.e., support *t* up to 3 only in most cases.

The motivation behind adaptive meta-heuristic-based *t*-way strategies is to overcome the parameter tuning problem of standard meta-heuristic based strategies. Such strategies introduce additional methods for automatic tuning of algorithmic parameters or search operators of the standard strategies. However, the challenge for adaptive strategies is performance degradation owing to the overhead of the additional methods for tuning. These methods, such as fuzzy logic further affect the performance of adaptive strategies.

2.6 Overview of Teaching Learning-based Optimization (TLBO) Algorithm

The adoption of TLBO since its appearance by researchers in a variety of domains proves its effectiveness and efficiency in the field of optimization. Following are some advantages of TLBO (Singh, Chaudhary et al. 2017, Gandomi and Kashani 2018, Wang, Li et al. 2018):

i. TLBO has no algorithm-specific parameters to tune for achieving good performance.

ii. TLBO is easy to implement.

iii. TLBO is computationally more efficient than other well-established meta-heuristic algorithms.

2.6.1 TLBO Variants and their Applications

Since its inception, many TLBO variants have been put forward to improve its performance. Apart from the original TLBO, the main TLBO variants available in the literature can be divided into three categories: modified-based, hybrid-based, and cooperative-based (see Figure 2.17). A discussion with some examples of each category is presented as follows.

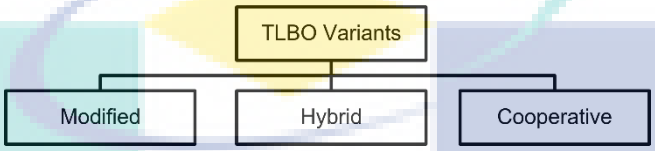


Figure 2.17 Types of TLBO Variants

As the names suggest, the modified-based category refers to variants that enhance TLBO's performance by modifying its parameter (e.g., elitism feature and adaptive behaviour) or altering the teacher and/or the learner phases. Rao and Patel (Rao and Patel 2012) introduced elitism feature within TLBO and demonstrate its efficiency for tackling 35 constrained benchmark functions. In other early work, Niknam et al. (Niknam, Azizipanah-Abarghooee et al. 2013) introduced an additional phase, termed modified phase, whereby four adaptive search operators are defined and probabilistically selected during runtime. The work has been successfully adopted for dynamic economic dispatch in power systems. Based on the same work, Amin et al. (Shabanpour-Haghighi, Seifi et al. 2014) also exploited the modified phase within TLBO and introduced an adaptive search operator based on Morlet wavelet function. With the fuzzy decision support (i.e., to select the best Pareto-optimal solution), the modified TLBO is then adopted for multi-objective optimal power flow problems. Although not introducing new phase, Hoseini et al. (Hoseini, Hosseinpour et al. 2014) adopted a similar approach for addressing multi-objective optimal location of automatic voltage regulators in the distribution system. Mandal and Roy (Mandal and Roy 2013) solved the multi-objective optimal reactive power dispatch problems by incorporating quasi-opposition based learning (QOBL) concept in the original TLBO algorithm to accelerate the convergence speed. In their work, Xia et al. (Xia, Gao et al. 2014) presented a modified TLBO for disassembly sequence planning problems. They modified the teacher–learner operator apart from introducing a feasible solution generator operator to satisfy the constraints of a disassembly sequence.

Most recently, Lei et al. (Lei, Gao et al. 2018) proposed teacher's teaching-learning-based optimization (TTLBO) algorithm for scheduling in hybrid flow shop to minimize energy consumption. The learner phase is replaced with self-learning of teachers and a crossover operator for global search. In other recent work, Niu et al. (Niu, Ma et al. 2018) proposed a modified TLBO called MTLBO for global optimization. MTLBO divides the learners into two groups based on the mean results in both the phases. The group of learners having best mean results increases their knowledge by interaction among themselves, whereas the group of learners with average mean results increases their knowledge by learning from their teacher. MTLB has shown better solution quality as well as faster convergence speed. Wang et al. (Wang, Li et al. 2018) proposed improved TLBO (ITLBO) for constrained optimization problems that modifies both the phases of TLBO. The teacher phase is divided into sub-population to enhance diversity, whereas the learner phase is based on the ranking differential vector to promote convergence.

Although producing sound results, modified-based TLBO algorithm is often applicable to specific problems and not sufficiently general (i.e., owing to problem domain assumption). As such, the performance of modified TLBO cannot be guaranteed even with the slight modification of the same problem instances.

Complementing the modified-based category, the hybrid-based category refers to the integration of one or more meta-heuristic algorithms (or their search operators) within TLBO. To date, TLBO has been used to form a hybrid model from many meta-heuristic algorithms. Jiang and Zhou (Jiang and Zhou 2013) explored the adoption of hybrid TLBO with differential evolution (DE) to solve the short-term optimal hydro-thermal scheduling. Tuo et al. (Tuo, Yong et al. 2013) implemented an improved harmony search based TLBO (HSTL) to balance between convergence speed and population diversity for general constrained optimization problems. Lim and Mat Isa (Lim and Isa 2014) integrated particle swarm optimization (PSO) with TLBO as an alternative strategy to cater for the local optimum problem within constrained benchmark functions. Recently, Huang et al. (Huang, Gao et al. 2015) integrated TLBO with the cuckoo search algorithm for the parameter optimization in structure designing and machining problem.

Indeed, while hybrid-based algorithm can be useful to capitalize on TLBO strengths and compensate on its deficiencies, the actual implementation can be bulky and

computationally heavy. Additionally, achieving a good balance between exploration and exploitation (of the hybrid search operators) can still be problematic.

Last but not least, the cooperative-based category refers to variants of TLBO that address large optimization problems with multiple-swarm populations. In this case, tasks are split into *k* sub-problems for simultaneous optimization before combining the results. Biswas et al. (Biswas, Kundu et al. 2012) highlighted the earliest work that exploits cooperative co-evolutionary TLBO with modified exploration strategy for large-scale optimization problems. Similarly, Satapathy and Naik (Satapathy and Naik 2013) explored cooperative TLBO (Co-TLBO) which allows cooperative behavior via the adoption of multiple swarm populations. In other work, Zou et al. (Zou, Wang et al. 2013) proposed the adoption of multiple swarm populations for the dynamic optimization problem.

Despite its potential, the key challenges of cooperative-based TLBO algorithm are twofold. The first challenge is to identify the best sub-problem size (and the multiple swarm populations). The second challenge is to model the independent variables to be placed in different sub-problems.

These and many other variants of TLBO suggest that its solution diversity and convergence speed can be improved further. This research work attempts to enhance original TLBO further by proposing a modified type variant that selects the appropriate phase in each iteration as per search requirements with the help of fuzzy logic. Moreover, neither original TLBO nor its variants solve the problem of *t*-way test suite generation before this work as per the review presented here.

2.7 Fuzzy Logic and Meta-heuristic Algorithms

Fuzzy logic (FL) encompasses fuzzy sets theory and possibility theory. The idea of FL was first coined by Zadeh in 1965 for representing and manipulating imprecise or fuzzy data. As a Soft Computing methodology, FL can effectively analyze complex systems, particularly when the data can be modeled with several linguistic parameters. Fuzziness is an essential feature of the language. The human brain is capable to interpret incomplete, vague or ambiguous sensory information accumulated by perceptive organs. Fuzzy set theory offers a systematic way to manipulate such information linguistically

and supports numerical computation through the use of linguistic labels which are specified by membership functions (Yen and Langari 1999).

In the recent literature, the use of FL for performance improvement of meta-heuristic algorithms appeared very effective. To date, two major forms of FL application in evolutionary and swarm-based optimization algorithms are: (a) performance enhancement by tuning parameters dynamically and (b) performance enhancement by hybridizing meta-heuristic algorithms (Ameli, Alfi et al. 2016). With the help of FL, these algorithms can obtain dynamic adaptation features. Dynamic tuning of parameters is a typical approach adopted for meta-heuristic algorithms to achieve further improvement. Here, a fuzzy-based system is employed with the goal for setting some parameters of a meta-heuristic algorithm (Neyoy, Castillo et al. 2013, Avila and Valdez 2015, Castillo, Meléndez et al. 2015, Pérez, Valdez et al. 2015, Solano-Aragón and Castillo 2015, Ameli, Alfi et al. 2016, Pérez, Valdez et al. 2017, Valenzuela, Valdez et al. 2017). For better performance and further optimal results, meta-heuristic algorithms have also been hybridized. Maintaining a balance between exploration and exploitation is required in hybrid algorithms so as to find acceptable solutions. Apart from parameter tuning and algorithm hybridization, FL has also been used with an algorithm to improve its performance (Cheng and Prayogo 2016). In this thesis, FL has applied in a new way as it selects either the exploration (i.e., global search) or the exploitation (i.e., local search) in the proposed adaptive TLBO (ATLBO).

2.8 Research Gap

The review of meta-heuristic-based strategies for *t*-way test suite generation and their categorization in previous sections served as useful tools to identify the research gap in the existing literature. Apparently, meta-heuristic algorithms appear suitable for *t*-way testing. However, a critical look unfolds some limitations of these algorithms as far as the complexity of both the algorithm structure and search process are concerned. For instance, although SA has been adopted by 15 out of 39 total reviewed strategies, it may need extensive computations owing to its update rule in the large random search space. This is especially true in case of either higher interaction strength (i.e., $t > 3$) or more complex system configuration (Cohen 2004, Afzal, Torkar et al. 2009). SA, being a single solution meta-heuristic, can be overly sensitive to its initial starting point in the search space, hence, suffers from early convergence. Moreover, the performance of SA depends

on properly scheduling the decrease in the temperature. Most SA based strategies, thus, are limited to small configurations with maximum interaction strength support of only $t = 3$.

TS (adopted by 8 strategies) also needs heavy computations in order to keep and update arrays in the *tabu list* (Ahmed, Zamli et al. 2012). TS requires more iterations in covering interaction tuples as compared to other meta-heuristic algorithms such as SA and PSO (Ahmed and Zamli 2011a). Similarly, no TS based strategy support the generation of VCAs. In GA, representation of problem via chromosomes can be troublesome. The crossover and mutation processes of GA make it computationally expensive. These processes slow down the array generation by GA as it takes more time than SA and TS while generating various CAs (Kuliamin and Petukhov 2011).

ACA, owing to its distributed nature, requires more computational power. Similarly, the complex algorithm structure of ACA limits its application for CAs and VCAs generation to smaller interaction strengths of $t = 2$ and $t = 3$. PSO (adopted by 6 strategies) despite of supporting interaction strength greater than 3, requires extensive parameter tuning before generating optimal results. Moreover, PSO suffers from other limitations such as falling in local minima and premature convergence which lead to poor optimization (Ahmed, Abdulsamad et al. 2015). HS-based strategy (Alsewari and Zamli 2012) not only generates CAs, MCAs, VCAs and CCAs but also supports higher interaction strength ($t = 15$). However, like PSO, HS has many parameters to tune prior to generating smallest arrays. Similarly, HS is computationally heavy owing to its frequent interaction with the environment. CS-based t-way strategy generated only CAs and MCAs for a small number of parameters and values with interaction strength support of up to $t = 4$. Although useful for t-way test suite generation, aggressive Lévy flight motion leads to poor exploitation in CS. Apart from the parameter tuning problem, BA suffers from the problem of early convergence and then followed by a slow convergence rate (Fister, Fister et al. 2014). Despite low algorithm complexity, it is difficult to divide the roles of bees as workers in the Bees Algorithm. Table 2.10 provides a comparison of the meta-heuristic-based strategies for the t-way test suite generation problem.

Based on the above discussion and comparison of meta-heuristic-based strategies for generating t-way test suites given in Table 2.10, it can be observed that:

Table 2.10 Existing Straetegies based on Meta-Heuristic Algorithms for *t*-way Test Suite Generation: Strengths and Weaknesses

Strategy	Strengths	Weaknesses
SA	• Adopted by most research studies in the literature • Offers most optimal *t*-way test suites • Generates CAs and all its variants	• Performance depends on its control parameters (i.e., initial temperature and cooling rate) • Computationally heavy particularly in case of complex configurations • Relies on standard structures to generate optimal test suites while it is difficult to have such structures
TS	• Supports configurations with many input parameters or configuration options • Avoids solutions that are already generated using the tabu list	• Needs tuning of its various parameters such as tabu list size, long term and short term memories • No support for generating mixed strength *t*-way test suites • Computationally heavy for large configurations
GA	• Supports generation test suites with highest interaction strength (i.e., $t = 20$). • Maintains a good balance between exploration and exploitation	• Needs to tune its different control parameters such as mutation rate, crossover rate and selection strategy • Computationally heavy owing to continuous interaction with peers and the environment
ACO	• Supports generation of both CAs and VCAs • Achieves optimal balance of exploration and exploitation	• Relies on several control parameters such as pheromone amount, pheromone coefficient, pheromone evaporation rate, etc. • Computationally heavy owing to its inherent parallel nature
PSO	• Offers optimal sizes CAs, MCAs, VCAs and CCAs • Supports generation of higher strength (i.e., $t \leq 6$) test suites	• Performance depends on many algorithm-specific parameters (i.e., social and cognitive parameters) • Frequent interaction with the environment as particles' velocities need to be updated continuously
HSS	• Supports generation of CAs, MCAs, VCAs and CCAs • Offers test suites with interaction strength support of $t = 15$	• Needs extensive tuning of its four control parameters • Computationally expensive because of the frequent interaction with the environment
CS	• Introduces Lévy flight motion to tackle entrapment in local optima • Supports generation of CAs and MCAs of competitive sizes with interaction strength support of $t = 6$	• Needs proper tuning of its single control parameter called switching probability • No support for generating mixed strength *t*-way test suites
BA	• Achieves a good balance between exploration and exploitation via unique features inherited from microbats • Offers smaller sizes CAs with interaction strength support of $t = 6$	• Relies on extensive tuning of its several control parameters • Tested on a limited set of benchmarks
Bees Algorithm	• Supports generation of both CAs and SCAs • Generates test suites of interaction strength support up to $t = 10$	• Relies on proper tuning of its six different control parameters • Faces challenge of assigning roles to worker bees

i. No single strategy based on meta-heuristic algorithm can generate optimal *t*-way test suite for all configurations. In line with the *No Free Lunch Theorem* (Wolpert and Macready 1997), this implies that the search for new and efficient strategies based on meta-heuristic algorithms, particularly newly developed ones, is still an active and open research topic.

ii. Most of the existing well-known strategies are not only computationally heavy but also require many parameters to be tuned for achieving good results. The number of tuned parameters, thus, needs strong consideration when developing new strategies.

iii. The adaptive meta-heuristic based strategies through dynamic tuning have been successfully explored. However, there is a lack of study on adopting *parameter-free* meta-heuristic algorithms such as TLBO.

iv. Although useful, TLBO has preset exploration and exploitation. For better performance, exploration and exploitation need to be dynamic.

The aim of this thesis is to fill these gaps by designing and implementing a *t*-way strategy based on an efficient and *parameter-free* meta-heuristic algorithm called adaptive TLBO (ATLBO) for addressing the problem of generating mixed strength *t*-way test suites. A Mamdani-type fuzzy inference system is integrated with the original TLBO in this research to adaptively select the search operations (global or local) on the basis of search need at that particular time of searching. ATLBO-based strategy is the first strategy based on a parameter-free meta-heuristic in the related literature for *t*-way test suite generation. Figure 2.18 summarizes the discussion given here.

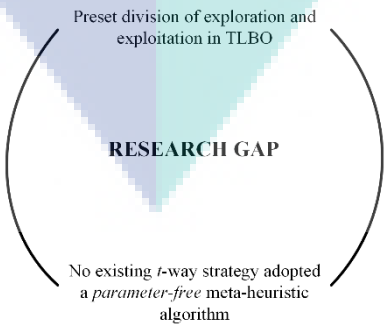


Figure 2.18 Research Problems in the Existing Related Literature

2.9 Chapter Summary

In essence, testers can use test case design techniques as a basis for the creation of test cases. Combinatorial *t*-way testing is complementing these techniques and has been proved effective in testing software applications with huge configuration spaces or input parameters. CAs and its variants abstract this type of testing by efficiently representing *t*-way test suites. As CAs generation is an optimization problem, several meta-heuristic based strategies have been proposed in the literature. These include *t*-way strategies based on meta-heuristic algorithms such as SA, TS, GA, ACA, PSO, HS, CS, BA and Bees Algorithm. Exiting strategies can be divided into three categories: standard, hybrid and adaptive. These strategies have produced best test suites (in terms of sizes) to date. However, all these strategies suffer from several problems such as extensive tuning of parameters, heavy computation, etc. before able to obtain optimal results. Moreover, only few strategies while no adaptive strategy in the literature generate VCAs of strength greater than 3.

Owing to these limitations of existing strategies, there is a need to propose an efficient strategy based on a newly developed *parameter-free* meta-heuristic algorithms for the generation of optimal test suites. TLBO is one such meta-heuristic algorithm. TLBO and its variants (modified, hybrid and cooperative) are found effective for the optimization problems in different fields of engineering and science. The algorithm offers demanding features: simple computational characteristics, ease of implementation and free of parameter tuning. Fuzzy set theory as a mathematical foundation of fuzzy logic has given computers the power to think and reason like humans. Fuzzy inference systems have been found effective for performance enhancement of meta-heuristic algorithms. Considering the features of original TLBO, an improved variant of TLBO called ATLBO based on the Mamdani-type fuzzy inference system, proposed in this research, is supposed to be effective for the problem of mixed strength test suite generation. Building upon this chapter, the next chapter presents the methodology adopted in this research for the design of ATLBO and its implementation for the mixed strength test suite generation as an effort to prepare optimal test suites.

CHAPTER 3

METHODOLOGY

In the previous chapter, a review of some well-known test case design techniques is presented. The importance of *t*-way and mixed strength interaction testing is elaborated using a simplified model of an online gaming architecture. Foundation terminologies and objects related to *t*-way interaction testing are defined. An extensive review of meta-heuristic based *t*-way strategies is given with their novel categorization. Teaching Learning-based Optimization (TLBO) algorithm and its variants with their applications are investigated. Fuzzy logic in the context of meta-heuristic algorithms is overviewed. Finally, research gap is identified in the existing literature so as to fill it with required new contributions.

This chapter starts by presenting an overview of the original TLBO with critically analyzing its searching capabilities. The design of the proposed ATLBO is presented next with detailed elaboration of its Mamdani-type fuzzy inference system. Hereafter, the chapter presents the pseudo code of the general ATLBO algorithm. After justifying the effectiveness of ATLBO, this chapter focuses on its implementation in a strategy to address the problem of mixed strength test suite generation. This chapter also discusses the implementation of *t*-way and mixed-strength interaction elements generation algorithm.

3.1 The Original Teaching Learning-based Optimization (TLBO) Algorithm

Teaching Learning-based Optimization (TLBO) (Rao, Savsani et al. 2011, Rao, Savsani et al. 2012) algorithm is a novel nature-inspired meta-heuristic algorithm for unconstrained and constrained optimization problems. In TLBO, the entire optimization process is equated with the teaching and learning methodology inside a classroom.

Students or learners are simulated as solutions whereas their subjects are represented as dimensions of the solutions (see Figure 3.1). The result of the class is regarded as the objective function value. TLBO exhibits competitive performances owing to its promising characteristics such as no algorithm-specific parameters, ease of implementation, computationally lightweight, and effective search ability (Singh, Chaudhary et al. 2017, Gandomi and Kashani 2018, Wang, Li et al. 2018).

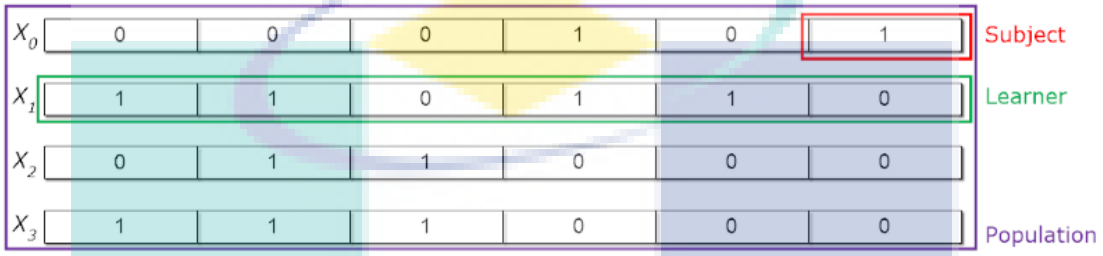


Figure 3.1 Concepts of TLBO for Optimization

The analogy adopted by TLBO from teaching and learning process between a teacher and his or her students or learners is further elaborated here. Basically, a teacher has more knowledge than the students. He or she tries to impart this knowledge to the students so as to take their knowledge to his/her competency level as shown in Figure 3.2 (a). As teachers have different competency levels, there could also be potential improvements when students learn from other teachers. At the same time, students can also learn from other students or peers with more knowledge to improve their competency levels as depicted in Figure 3.2 (b).

Within TLBO, the solution is represented in the population X. An individual X_i within the population represents a single possible solution. Specifically, X_i is a vector with D elements where D is the dimension of the problem representing the subjects taken by the students or taught by the teacher.

TLBO divides the whole searching process into two main phases; the teacher phase and the learner phase. In order to perform the search, TLBO undergoes both phases sequentially one-after-the-other per iteration. The teacher phase involves invoking the global search operation (i.e., exploration). At any instance of the search process, the teacher is always assigned to the best individual X_i. The algorithm attempts to improve other individual X_i by moving their position towards $X_{teacher}$ taking into account the current mean value of the population, X_{mean} as follows:

$$X_i^{t+1} = X_i^t + r(X_{teacher} - T_F X_{mean}) \quad 3.1$$

where X_i^{t+1} is the new updated X_i^t, $X_{teacher}$ is the best individual in the population X, X_{mean} is the mean of X, r is the random number from [0,1] and T_F is a teaching factor which can either be 1 or 2 and meant for emphasizing the quality of teaching. It is tested with various values but TLBO is more successful when it is either 1 or 2 (Chikh, Belaidi et al. 2018).

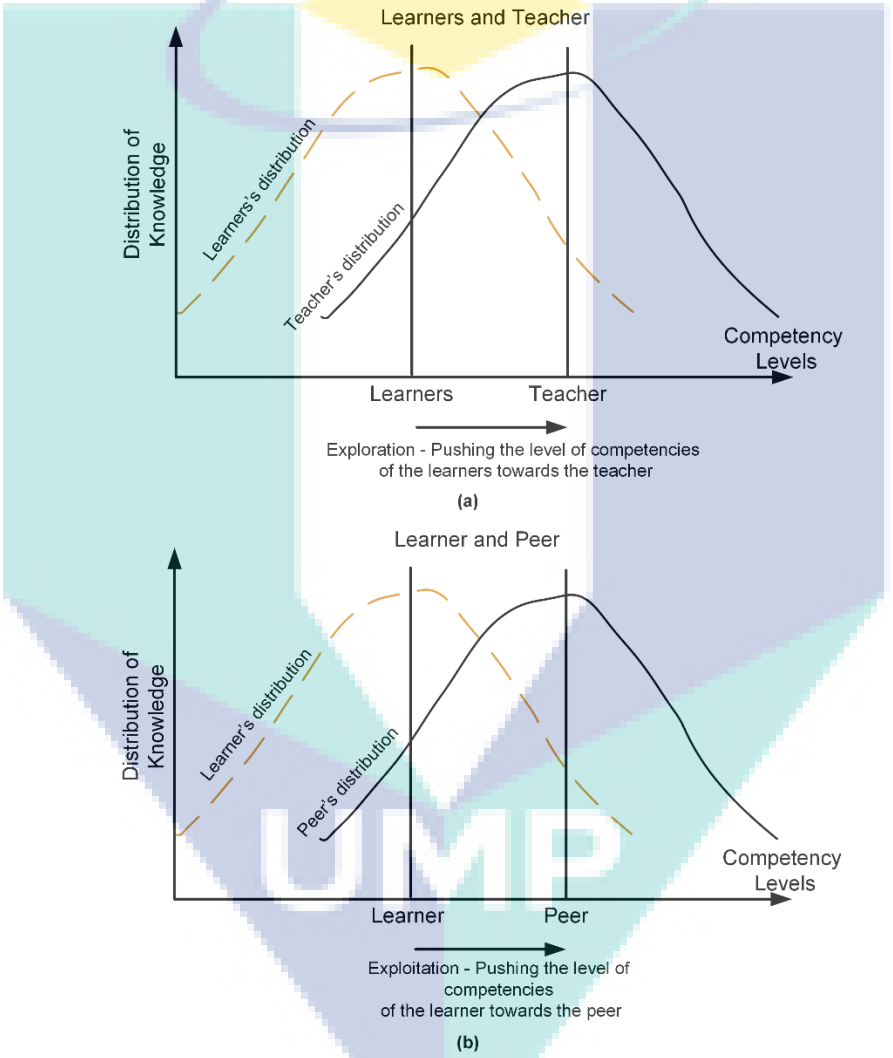


Figure 3.2 TLBO's Teaching and Learning Analogy

Source: (Rao, Savsani et al. 2011)

The learner phase exploits the local search operation (i.e., exploitation). Specifically, the learner X_i^t increases its knowledge by interacting with its random peer X_j^t within the population X such that $i \neq j$. A learner learns if and only if the other learner

has more knowledge than he does. At any iteration i, if X_i^t is better than X_j^t, then X_j^t moves toward X_i^t (refer to Eq. 3.2). Otherwise, X_i^t moves toward X_j^t (refer to Eq. 3.3).

$$X_i^{t+1} = X_i^t + r\,(X_j^t - X_i^t) \tag{3.2}$$

$$X_i^{t+1} = X_i^t + r\,(X_i^t - X_j^t) \tag{3.3}$$

where X_i^{t+1} is the new updated X_i^t , X_j^t is the random peer, and r is the random number from [0,1]. The original TLBO can be summarized in Figure 3.3.

The Original TLBO Algorithm

Input: The population $X = X_1, X_2 \ldots X_D$
Output: X_{best} and the updated population $X' = \{X'_1, X'_2, \ldots, X'_D\}$

1 Initialize random population of learners X and evaluate them
2 **while** *stopping criteria not met* **do**
3 **for** $i = 1$ *to population size* **do**
  /* Teacher Phase ... Exploration */
4 Select $X_{teacher}$ and calculate X_{mean}
5 $T_F = round(1 + r(0,1))$
6 $X_i^{t+1} = X_i^t + r(X_{teacher} - T_F X_{mean})$
7 **if** $f(X_i^{t+1})$ *is better than* $f(X_i^t)$ **then**
8 $X_i^t = X_i^{t+1}$
9 **end**
  /* Learner Phase ... Exploitation */
10 Randomly select one learner X_j^t from the population X such that $i \neq j$
11 **if** $f(X_i^t)$ *is better than* $f(X_j^t)$ **then**
12 $X_i^{t+1} = X_i^t + r(X^t{}_j - X^t{}_i)$
13 **end**
14 **else**
15 $X_i^{t+1} = X_i^t + r(X^t{}_i - X^t{}_j)$
16 **end**
17 **if** $f(X_i^{t+1})$ *is better than* $f(X_i^t)$ **then**
18 $X_i^t = X_i^{t+1}$
19 **end**
20 **end**
21 Get best result X_{best}
22 **end**

Figure 3.3 The Original TLBO Algorithm

In line 1, the algorithm initializes a random population of learners and evaluates them. Line 2 starts the main loop of the algorithm where the termination condition is specified. The termination condition can be the number of iterations, objective function evaluations, etc. In the case of *t*-way test suite generation problem, the termination condition is the coverage of all the interaction tuples or elements. Line 3 starts the sub-

loop which is repeated till the evaluation of all the learners. The algorithm selects the best candidate solution (i.e., the teacher) in the population and calculates the population mean in line 4. With this, the algorithm commences the teacher phase. The teaching factor T_F is computed in line 5. The main equation of the teacher phase (Eq. 3.1) is evaluated in line 6 which attempts to enhance the learning capabilities of each learner in the population through teaching. Finally, the learner's knowledge is checked and subsequently updated in the case of improvement (lines 7-9). Lines 4-9 constitute the teacher phase.

The learner phase follows the teacher phase immediately. Line 10 starts the learner phase. A learner is randomly selected from the population such that its roll number does match with the roll number of the current learner. If the knowledge level of the current learner is better than the newly selected learner, Eq. 3.2 is evaluated in line 12, otherwise Eq. 3.3 is computed in line 15. Here, a new learner with some knowledge level comes whether Eq. 3.2 or Eq. 3.3 is evaluated. Lines 17-19 update the knowledge of the current learner if his knowledge is poor than the new learner. Lines 10-19 constitute the learner phase. Line 20 ends the sub-loop after evaluating all the learners. A best so far learner (i.e., solution) is returned by the algorithm in line 21. These steps are repeated until the satisfaction of the termination criteria specified in the main loop. The final X_{best} is the global optimum solution offered by TLBO. Line 22 ends the algorithm.

3.2 The Proposed Fuzzy Adaptive TLBO (ATLBO)

This section presents the general design of the proposed fuzzy adaptive TLBO (ATLBO). The section first briefly describes the Mamdani-type fuzzy inference system of ATLBO and then explains the ATLBO algorithm by presenting its pseudo code.

3.2.1 The Mamdani-type Fuzzy Inference System of ATLBO

A fuzzy inference system is often defined as a system that represents human knowledge mapped in the form of a set of fuzzy rules to produce some approximate decision (Iancu 2012). Classical logic (crisp value) is differentiated from fuzzy sets by the introduction of membership functions. According to Zadeh (Zadeh 1965), $\mu(A(x))$ denotes the membership of element x of the fuzzy set A, and can gradually transform from $\mu(A(x)) = 1$ (full membership) to $\mu(A(x)) = 0$ (full non-membership). Consequently, it can take all possible values from the interval [0, 1], and is therefore not restricted to only two truth values (0 or 1) of classical logic. This thesis adopted a Mamdani-type

fuzzy inference system (Mamdani and Assilian 1975, Cordón 2011, Camastra, Ciaramella et al. 2015) for the proposed fuzzy adaptive TLBO (ATLBO) owing to its effectiveness for performance enhancements of meta-heuristic algorithms in the related literature.

The methodology of the Mamdani fuzzy inference systems captures the process behaviours by using linguistic variables and then uses these variables as input to the linguistic control rules (Dadios 2012). The fuzzy model based on the Mamdani fuzzy inference systems involves defining membership functions and subsequently describing the rules. The rules act as a bridge between input and output variables and are based on the description of the fuzzy behaviour that is obtained by defining linguistic variables (Zimmermann 1996).

A Mamdani fuzzy inference system encompasses the fuzzification, rule base with its inference system and defuzzification as its basic components. The general structure of a Mamdani fuzzy inference system, however, include the membership functions, input/output variables and the rules. The Mamdani-type fuzzy inference system designed for ATLBO is shown in Figure 3.4.

The fuzzy system is composed of three components, namely the fuzzification, fuzzy rules inference evaluation, and defuzzification (refer to Figure 3.4). There are three input parameters and one output parameter of the system. The three input parameters are: *Quality Measure* (Q_m), *Intensification Measure* (I_m), and *Diversification Measure* (D_m) and the one output parameter is: *Selection*.

The choice of selecting and using these three input parameters is new for improving the performance of a meta-heuristic algorithm. These parameters capture all the necessary details related to a potential solution so as to achieve optimality by guiding the search in the right direction. The Q_m takes into account the number of interaction elements covered by a candidate test case. The other two parameters (I_m and D_m) are based on Hamming distance. The quality and diversification measures are used to achieve solution diversity, whereas the intensification measure is used to facilitate convergence.

The first input parameter or crisp input Q_m is the normalized fitness value capturing the quality of the current potential solution $X_{current}$ in terms of covering the

number of interaction tuples or elements. The value of the Q_m will be high for a test case that covers a maximum number of interaction tuples. Eq. 3.4 formally defines Q_m.

$$Q_m = \left[\frac{X_{current\ fitness} - min\ fitness}{max\ fitness - min\ fitness}\right] \cdot 100 \quad 3.4$$

The second parameter or crisp input I_m is the Hamming distance normalized value measuring the proximity of the $X_{current}$ against X_{best}.. I_m can be formally defined as:

$$I_m = \left[\frac{|X_{best} - X_{current}|}{D}\right] \cdot 100 \quad 3.5$$

where D is the dimension of the given problem.

The third and final crisp input D_m is also the Hamming distance normalized value. Unlike I_m which measures intensification of the search against the global best, D_m measures diversity of $X_{current}$ against the overall population X. D_m can be formally defined as follows:

$$D_m = \left[\frac{\sum_{j=1}^{population\ size} |X_j - X_{current}|}{D}\right] \cdot 100 \quad 3.6$$

First, fuzzification starts the fuzzy inference process by transforming the crisp inputs into fuzzy inputs with the help of membership functions. All crisp inputs have universes of discourse, i.e., set of possible values that crisp inputs can assume. The universes of discourse for all the linguistic variables of the proposed fuzzy inference system are between 0 and 100. The fuzzification process of all the input variables is based on three defined trapezoidal membership functions with linguistic terms, namely *Low*, *Medium* and *High* (see Figure 3.5Figure 3.5). It is worth noting that the trapezoidal membership functions for the linguistic variables Quality Measure and Diversification Measure are identical as shown in Figure 3.5 (a) and Figure 3.5 (c). Similarly, the fuzzification process of the output variable *Selection* is based on two defined trapezoidal membership functions with linguistic terms, namely *Global_Search* and *Local_Search* as shown in Figure 3.6.

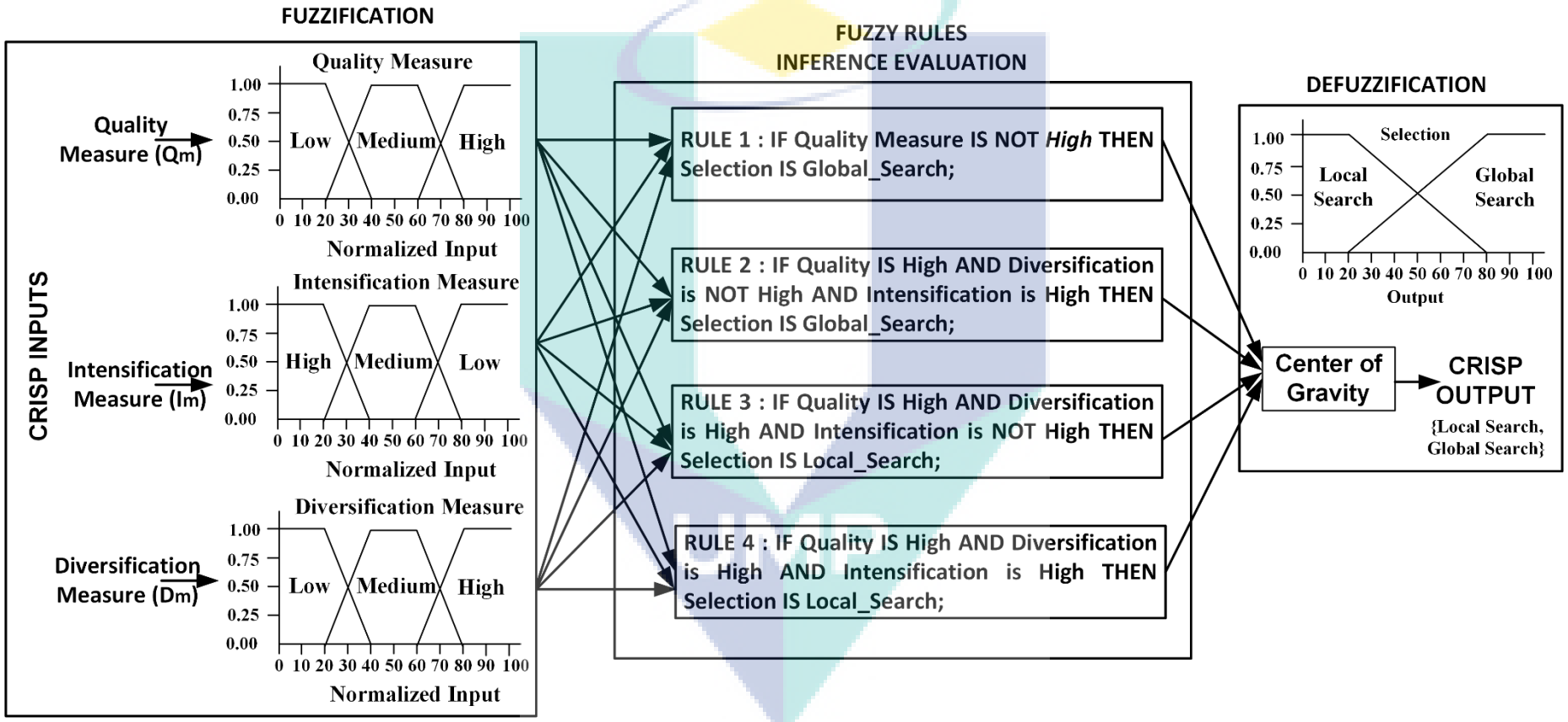


Figure 3.4 Fuzzy Inference System for ATLBO

The values in the range of 0-20 are considered as absolute *Low*; the values in the range of 20-40 are considered as partial *Low* and *Medium*; the values in the range of 40-60 are considered as absolute *Medium*; the values in the range of 60-80 are considered as partial *Medium* and *High*; the values in the range of 80-100 are considered as absolute *High*. The *High* and Low ranges in the case of Intensification Measure are exchanged (refer to Figure 3.5 (b)). Conversely, change did not occur in the *Medium* range. The *Selection* values taking the range of 0-20 are considered as absolute *Local_Search*; the *Selection* values in the range of 20-80 are considered as partial *Local_Search* and *Global_Search*; the *Selection* values in the range of 80-100 are considered as absolute *Global_Search*.

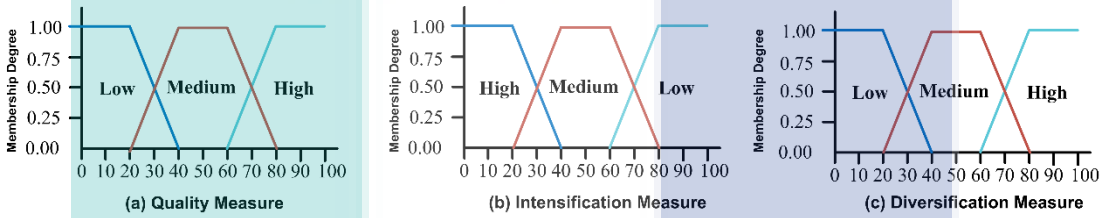


Figure 3.5 Membership Functions of the three Input Measures

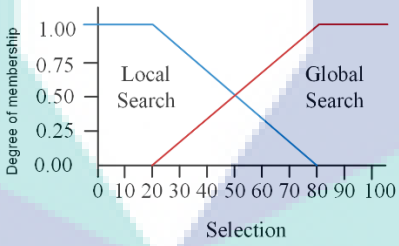


Figure 3.6 Membership Functions of the Selection Output Linguistic Variable

Decision making logic is the next step after finalizing the membership functions for fuzzy inputs and output of the system. The Mamdani-type fuzzy inference evaluation of the proposed system performs this step, where IF-THEN rules contain fuzzy prepositions in both IF (known as antecedent or premise) and THEN (known as consequent or conclusion) parts (Iancu 2012). An *i*th rule in the Mamdani-type fuzzy inference system can be written as follows:

$$\text{Rule } i\text{th: IF x IS } A_i \text{ AND y IS } B_i \text{ THEN O IS } C_i \qquad 3.7$$

where x and y denote input linguistic variables, O denotes the output variable linguistic variable, and A_i, B_i, and C_i denote the linguistic terms defined for the fuzzy variables.

The fuzzy rule base of ATLBO consists of only four fuzzy rules as shown in Table 3.1. These rules guide ATLBO to select the appropriate search operation. The total number of fuzzy rules *r* for a fuzzy inference system is calculated using Eq. 3.8.

Table 3.1 Fuzzy Rule Base of the ATLBO Fuzzy Inference System

Rule #:	**Rule**
Rule 1:	IF Quality IS NOT High THEN Selection IS Global_Search
Rule 2:	IF Quality IS High AND Diversification IS NOT High AND Intensification IS High THEN Selection IS Global_Search
Rule 3:	IF Quality IS High AND Diversification IS High AND Intensification IS NOT High THEN Selection IS Local_Search
Rule 4:	IF Quality IS High AND Diversification IS High AND Intensification IS High THEN Selection IS Local_Search

$$\mathrm{r} = \prod_{i}^{n} f_i \qquad 3.8$$

where n is the total number of crisp variables and f_i is the number of linguistic terms defined for each input linguistic variable.

In the case of ATLBO's fuzzy inference system, there can be a total of 3^3 or 27 rules as each input linguistic variable takes three linguistic terms. However, it is observed that the rules can be reduced to four only as shown in Table 3.1. Minimum possible rules not only simply the processing logic but may also improve the fuzzy system performance.

Concerning the fuzzy rules inference evaluation, the fuzzy rules are defined based on the following scenarios:

- Rule 1: Quality measure is *Low* regardless of intensification and diversification measures. The search is trapped in the local minima region, thus requiring global search.
- Rule 2: Quality measure is *High* but lacks diversity. The search is trapped in the local minima region because of excessive local search.
- Rule 3: Quality measure is *High* but lack of convergence because of excessive global search.

- Rule 4: Search is near convergence. Local search is required.

The max-min inference method is adopted in the proposed fuzzy system. The method interprets the fuzzy operator AND by considering the minimum value of the antecedents while adopting maximum value for aggregating them (see Figure 3.7).

Finally, the defuzzification step transforms the fuzzy conclusions of the inference scheme into the crisp output. As described earlier, a single output linguistic variable called *Selection* is defined for the defuzzification. Eventually, the actual selection depends on the output of the deffuzzification process based on the center of gravity (COG). COG is the most commonly adopted method in fuzzy systems owing to its accurate computation of results on the basis of weighted values of many output membership functions (Pappis and Siettos 2014). The result of defuzzification is assigned to the *Selection* crisp variable after the evaluation of COG formula according to Eq. 3.9.

$$Selection = COG = \begin{cases} \dfrac{\int_x \mu(A(x)).x\,dx}{\int_x \mu(A(x)).dx} & ; \text{if x is continous} \\ \dfrac{\sum_x \mu(A(x).x}{\sum_x \mu(A(x))} & ; \text{if x is discrete} \end{cases} \qquad 3.9$$

where $\mu(A(x))$ denotes the membership function value of the output fuzzy set.

Suppose quality, intensification and diversification parameters have values 65, 70 and 80, respectively, as shown in Figure 3.7.

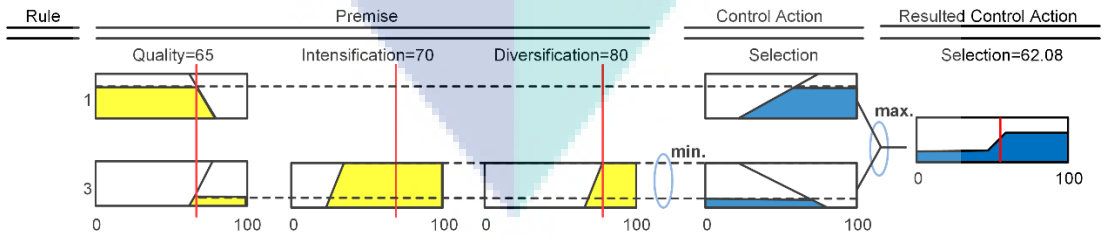


Figure 3.7 Max-min Inference Method and Defuzzification

With Q_m= 65, rule # 1 will be activated as Quality Measure is only partially associated with the fuzzy set High. Similarly, rule # 3 will be fired on the subset partial

High, partial Medium and Low, and absolute High of input linguistic variables Quality Measure, Intensification Measure and Diversification Measure, respectively. By using Eq. 3.9, the defuzzification step can imply the crisp output of the fuzzy inference system for the selection variable as *Selection* = 62.08.

A number of design choices such as triangular memberships, the number of linguistic terms, etc., have been relevant in the implementation of the fuzzy inference system of ATLBO. However, the proposed fuzzy inference system is easy to understand, functional and sufficiently efficient owing to the adoption of the basic design choices.

It is evident from the overview of the original TLBO where both global search and local search operations get equal opportunity (50%) in each iteration during the search process. Therefore, when the defuzzification output of the fuzzy system assigned to the S*election* crisp out is greater than 50%, the proposed algorithm selects global search or teacher phase. Otherwise, it selects local search or learner phase.

3.2.2 The General ATLBO Algorithm

Based on the proposed fuzzy inference system along with the TLBO description given in the previous section, Figure 3.8 highlights the newly developed adaptive TLBO (ATLBO) based on the TLBO description provided in the previous section. The boxes mark where ATLBO code differs from original TLBO code.

Line 1 defines trapezoidal membership functions for the three input parameters and one output parameter to obtain the linguistic variables. Line 2 defines the four fuzzy rules as explained previously. The algorithm initializes a random population of learners and evaluates them in line 3. Line 4 sets the *Selection* variable to 100 in order to run teacher phase in the first iteration similar to the original TLBO. Line 5 starts the main loop of the algorithm. Line 6 starts the for loop so as to search all learners and improve their competency levels in either teacher phase or learner phase. Line 7 checks whether the *Selection* variable is greater than 50. Based on this condition, the algorithm either runs teacher phase (lines 8-13) or learner phase (lines 16-25).

The General Adaptive TLBO (ATLBO) based on Fuzzy Inference System

Input: The population $X = X_1, X_2 \ldots X_D$
Output: X_{best} and the updated population $X' = \{X'_1, X'_2, \ldots, X'_D\}$

1 Define membership functions for the linguistic variables
2 Define fuzzy rules
3 Initialize random population of learners X and evaluate them
4 Set *Selection* = 100
5 **while** *stopping criteria not met* **do**
6 **for** $i = 1$ *to population size* **do**
7 **if** *Selection* > 50 **then**
  /* Teacher Phase ... Exploration */
8 Select $X_{teacher}$ and calculate X_{mean}
9 $T_F = round(1 + r(0,1))$
10 $X_i^{t+1} = X_i^t + r(X_{teacher} - T_F X_{mean})$
11 **if** $f(X_i^{t+1})$ *is better than* $f(X_i^t)$ **then**
12 $X_i^t = X_i^{t+1}$
13 **end**
14 **end**
15 **else**
  /* Learner Phase ... Exploitation */
16 Randomly select one learner X_j^t from the population X such that $i \neq j$
17 **if** $f(X_i^t)$ *is better than* $f(X_j^t)$ **then**
18 $X_i^{t+1} = X_i^t + r(X^t{}_j - X^t{}_i)$
19 **end**
20 **else**
21 $X_i^{t+1} = X_i^t + r(X^t{}_i - X^t{}_j)$
22 **end**
23 **if** $f(X_i^{t+1})$ *is better than* $f(X_i^t)$ **then**
24 $X_i^t = X_i^{t+1}$
25 **end**
26 **end**
27 Compute Q_m, I_m and D_m
28 Fuzzify based on Q_m, I_m and D_m
29 Defuzzify and set *Selection* = crisp output
30 **end**
31 Get best result X_{best}
32 **end**

Figure 3.8 ATLBO based on Fuzzy Inference System

Line 27 computes the three crisp measures or input parameters (Q_m, I_m, and D_m) for the current obtained solution. Line 28 fuzzifies these crisp measures using simple membership functions. After inference evaluation process, line 29 performs defuzzification to obtain crisp output and assigns it to the *Selection* variable. The value of the *Selection* variable will decide whether to run the teacher phase or the learner phase in the next for loop iteration. In line 31, the algorithm returns the current best result.

3.3 Computation of the Measures for *t*-way Testing

This section describes how the three measures (Q_m, I_m and D_m) are computed with an example in case of *t*-way testing as the case study. For this purpose, the online gaming architecture model discussed in Chapter 2, Section 2.2 is reconsidered as shown in Figure 3.9. It is a system with 5 parameters where 3 parameters carry 2 values whereas 2 parameters carry 2 values.

Maximum fitness (*max_fitness* in Eq. 3.4) is equal to the value of Eq. 2.2. For the running example, *max_fitness* is 10. Minimum fitness (*min_fitness* in Eq. 3.4) is 0. Current fitness (*current_fitness* in Eq. 3.4) is calculated in terms of interaction tuples' coverage. For instance, the Q_m for a test case $tc_1 = \{0\ 1\ 0\ 2\ 1\}$ is computed below if it covers 9 interaction tuples (i.e., it current fitness is 9). The value 90 is the normalized value for the fuzzy inference system.

$$Q_m = \left[\frac{X_{current\ fitness} - min\ fitness}{max\ fitness - min\ fitness}\right] \cdot 100$$

$$= \left[\frac{9-0}{10-0}\right] . 100 = 90$$

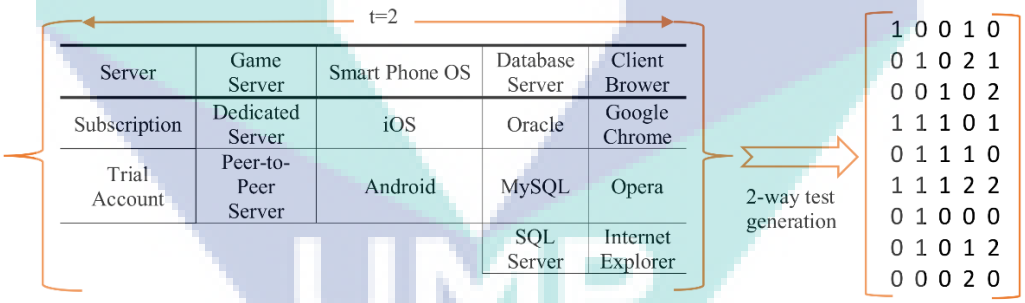


Figure 3.9 Pairwise Test Suite for the Online Gaming Architecture

As far as the intensification measure (I_m) is concerned, it is the Hamming distance between the current test case and the best case in the population divided by the vector *D*, which is equal to *P*, the total number of parameters. If $tc_2 = \{1\ 1\ 1\ 0\ 1\}$ is considered as the current test case and $tc_3 = \{1\ 1\ 1\ 2\ 2\}$ is the best case in the population as shown in Figure 3.10, the Hamming distance for this case is 2 as all the values of both the test cases differ by 2. The I_m for this case is (which is its normalized value):

$$I_m = \left[\frac{|X_{best} - X_{current}|}{D}\right] \cdot 100$$

$$= \left[\frac{2}{5}\right].100 = 40$$

Finally, diversification measure (D_m) is the Hamming distance between a current test case and the entire population divided by D which is computed using Eq. 3.10.

$$D = (Population_{Size} - 1) * P \quad 3.10$$

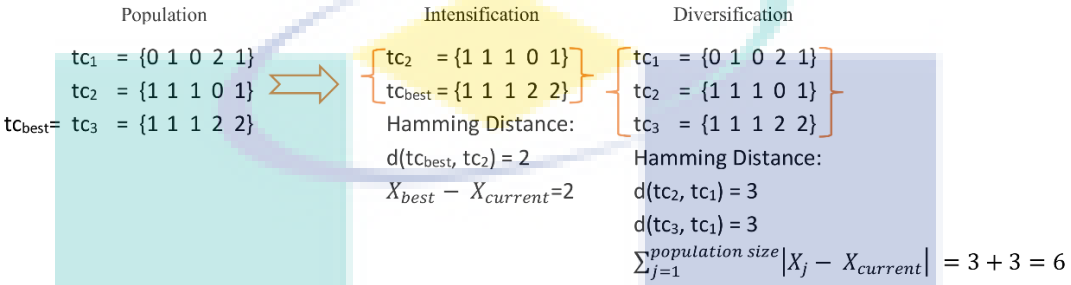


Figure 3.10 Hamming Distance Calculation for Intensification and Diversification

The population size for the current example is 3, whereas P is 5. If tc_1={0 1 0 2 1} is considered as the current test case from Figure 3.10, then the Hamming distance between tc_1 and tc_2 is 3 and the same between tc_1 and tc_3 is 3. The accumulative Hamming distance from the rest of the population for tc_1 is 3+3 = 6 (see Figure 3.10) as it differs by 2 values from tc_2 and tc_3. The normalized value of D_m is computed as follows:

$$D_m = \left[\frac{\sum_{j=1}^{population\ size} |X_j - X_{current}|}{D}\right] \cdot 100$$

$$= \left[\frac{6}{10}\right].100 = 60$$

3.4 Implementation of ATLBO for the Mixed Strength *t*-way Test Suite Generation

Having given an overview of TLBO and its adaptive variant (ATLBO), the following section outlines its application to address the problem of generating mixed strength *t*-way test suite. In general, ATLBO based strategy is a composition of two main algorithms: (i) an interaction elements generation algorithm, which generates combinations of parameter values that are used in the test suite generator for optimization purposes; (ii) an ATLBO based test suite generator algorithm. The next sub-sections, explain these two algorithms in detail.

3.4.1 Interaction Elements Generation Algorithm

The interaction elements generation algorithm involves generating the parameter *(P)* combinations and the values *(v)* for each parameter combination based on the interaction strength *(t)*. The parameter generation adopts binary digits, whereby 0 indicates the exclusion of a referred parameter and 1 indicates the inclusion of the parameter.

As an illustration, consider an example involving $VCA(N; 2, 2^3 3^1, CA(3; 2^3))$ as shown in Figure 3.11.

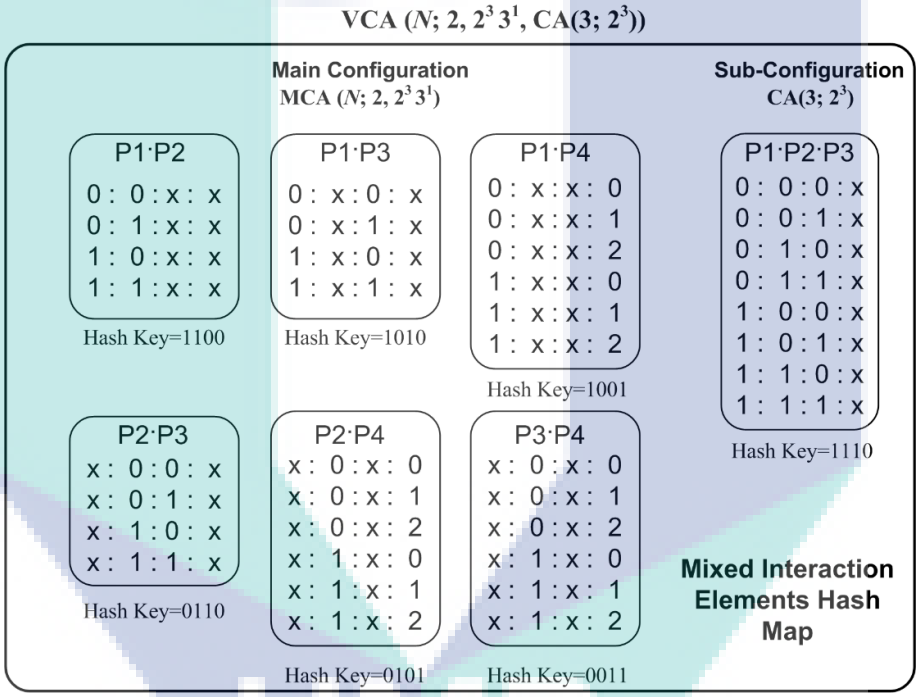


Figure 3.11 The Hash Map and Interaction Elements for the VCA

The mixed strength covering array VCA can be composed of two parts: the main configuration $MCA(N; 2, 2^3 3^1)$ and the sub-configuration $CA(3; 2^3)$, respectively. The main configuration, $MCA(N; 2, 2^3 3^1)$, requires a *2*-way interaction (as main strength) for a system of four parameters. The algorithm first generates all binary number possibilities up to four digits. Subsequently, the binary numbers that contain two 1 s are selected, indicating that a pairwise interaction (i.e., $t = 2$) exists. For example, the binary number 1100 refers to $P_1.P_2$ interaction. P_1 has two values (0 and 1); P_2 has two values (0 and 1); P_3 has two values (0 and 1), and P_4 has three values (0, 1, and 2). The *2*-way parameter interaction has six possible combinations based on the parameter generation algorithm.

For combination 1001, whereby P_1 and P_4 are available, there are 2×3 possible interaction elements between P_1 and P_4. For each parameter in the combination (i.e., with two 1 s), the value of the corresponding parameter is included in the interaction elements. Here, the excluded values are marked as "x". This process is iteratively repeated for the other five interactions: (P_1, P_2), (P_1, P_3), (P_2, P_3), (P_2, P_4), and (P_3, P_4). In a similar manner, the sub-configuration, CA (3; 2^3), requires a *3*-way interaction (as sub-strength) for a system of 3 parameters. A *3*-way interaction yields the (P_1, P_2, P_3) interaction.

Revisiting the overall *VCA(N; 2, 2^3 3^1, CA(3; 2^3))*, the complete interaction elements are the combinations from both *MCA(N; 2, 2^3 3^1)* and *CA(3; 2^3)*. The hash map list of mixed interaction elements H_s, which employs the binary representation of the interaction as the map retrieval key, is implemented to ensure efficient indexing for storage and retrieval. The complete algorithm for the interaction elements generation is highlighted in Figure 3.12.

Algorithm for Interaction Elements Generation

Input: parameters' values p, strength of coverage t, sub strength of coverage t_{sub}
Output: H_s representing the hash map list of $t-interaction$ elements

```
1  Initialize H_s = ∅
2  Let m = max no. of defined parameters
3  Let p = {p_0, p_1, ..., p_j} where p is the set of values for each parameter
4  for index = 0 to 2^m − 1 do
5      Let b = binary number
6      b = convert index to binary
7      for i = 0 to length of b do
8          if b[i] is equal to "1" then
9              Add the representative interaction value for p[index]
10             else
11                 Add don't care value for p[index]
12             end
13         end
14     end
15     if (the no. of 1's in b = t) or (the no. of 1's in b = t_sub) then
16         Set hashkey = b
17         Put p into the hashmap H_s using the hashkey
18     end
19 end
20 Return H_s
```

Figure 3.12 Algorithm for Interaction Elements Generation

Line 1 initializes an empty hash map H_s. Line 2 sets variable *m* to the number of parameters. The set of values of parameters are stored in a data structure called *p* (see line 3). The main loop of the algorithm starts at line 4 which runs 2^m times. The body of this loop generates interaction elements between parameters based on the given interaction strength *t* along with the generation of their hash key in the hash map. A binary type array *b* is declared to hold the binary form of the *index* variable (lines 5-6). The sub-

loop (lines 7-14) generates the interaction elements to be added to the hash map. The hash key for the added interaction element is inserted into the hash map (see lines 15-18). Line 20 returns the hash map containing all the interactions elements for a given configuration.

3.4.2 Test Suite Generation Algorithm based on ATLBO

ATLBO first initializes the population search space as a D-dimensional vector, $X_j = [X_{j,1}, X_{j,2}, \ldots\ldots\ldots, X_{j,D}]$, where each dimension represents a parameter and contains integer numbers between 0 and (v_i) (i.e., the number of values of the i^{th} parameter). TLBO requires both local search and global search to be summoned per iteration. Conversely, ATLBO permits the adaptive selection of the local search and global search through the fuzzy inference selection. The net effect is that ATLBO has less fitness function evaluations than the original TLBO for the same number of iterations.

For the mixed strength test suite generation problem, this research deals with the discrete version of ATLBO. As such, to deal with discrete parameters and values, each individual X_j needs to capture the parameters as a valid range of integer numbers (i.e., based on the user inputs). Local search and global search updates in ATLBO may result into necessary rounding off of floating point values.

The rounding off of floating point values should be addressed, as well as the out-of-range values. The *clamping rule* at the boundary within ATLBO is established to restrict both lower and higher bounds. At least three possibilities exist in dealing with boundary conditions used in the literature for discrete problems, i.e., invisible walls, reflecting walls, absorbing walls (Robinson and Rahmat-Samii 2004). In the invisible walls, when a current value goes outside the boundary, the corresponding fitness value is not computed. In the reflecting walls, when a value reaches the boundary, it is reflected back to the search space (i.e., mirroring effects). The boundary condition returns the current value to the search space when the value moves out-of-range in the absorbing walls. For example, if a parameter value is in the range from 1 to 4, the position is reset to 1 when it reaches a value larger than 4. In this study, the absorbing walls approach is used as the clamping rule in the implementation of ATLBO.

The local and global search processes of ATLBO are iteratively continued until convergence has been achieved (i.e., if and only if all the interaction elements from the H_s are completely removed), in relation to the stopping criteria.

Two approaches are considered for storing and locating the interaction elements: array list and hash map. The array list approach is fast for a small number of values but is not scalable for large parameters, because it must iterate the entire lists to fetch the required interaction values. Given that the process of fetching and locating the required interaction values are fundamentally important for fitness function evaluation, the array list approach can introduce time performance penalty. Alternatively, the hash map offers an effective approach of locating the required interaction values using only the unique key based on the binary interaction value itself. For this reason, the hash map approach is favored for the ATLBO. The ATLBO test suite generator is summarized in Figure 3.13 based on the aforementioned design choices.

Line 1 calls the interaction elements generation algorithm to enumerate the interaction elements in the hash map H_s based on the interaction strength, the number of parameters and their values. Line 2 defines the trapezoidal membership functions for the input and output linguistic variables. Line 3 defines the fuzzy rules of the fuzzy inference system. Line 4 initializes a random population of learners and evaluates them. Lines 5 is meant to run the teacher phase in the first iteration. The algorithm enters into its main loop in line 6 which will terminate on coverage of all the interaction elements or tuples. The algorithm enters into the for loop in line 7 to update the members of the population using the teacher or learner search operators. As the *Selection* variable is set to 100 (line 8), the algorithm runs the teacher phase in the first iteration (see lines 9-14). Line 28 computes the three measures for the current best test case. Line 29 converts the crisp input into fuzzy input. Line 30 defuzzify the linguistic variables by using the COG defuzzification method and assigns the crisp output to the *Selection* variable. ATLBO uses the value of this variable in the next iteration to run either the teacher phase (lines 9-14) or the learner phase (lines 17-27). Line 32 accumulates the best test case and stores it in the final test suite list F_s. The algorithm then removes the interaction elements or tuples from the hash map H_s covered by the current test case in line 33. Finally, line 35 displays the obtained optimal test suite F_s.

Applying ATLBO for Mixed Strength t-way Test Suite Generation

Input: Parameters' p values v, strength of coverage t, and sub strength of coverage t_{sub}
Output: The final test suite F_s

Initialize the required $t-way$ interaction elements in the hash map H_s based on p, v, t, and t_{sub}
Define the membership functions for the linguistic variables
Define the fuzzy rules
Initialize random population of learners X and evaluate them
Set $Selection = 100$
while *the hash map* H_s *is not empty* **do**
    for $i = 1$ *to population size* **do**
        if $Selection > 50$ **then**
            `/* Teacher Phase ... Exploration */`
            Select $X_{teacher}$ and calculate X_{mean}
            $T_F = round(1 + r(0,1))$
            $X_i^{t+1} = X_i^t + r(X_{teacher} - T_F X_{mean})$
            if $f(X_i^{t+1})$ *is better than* $f(X_i^t)$ **then**
                $X_i^t = X_i^{t+1}$
            end
        end
        else
            `/* Learner Phase ... Exploitation */`
            Randomly select one learner X_j^t from the population X such that $i \neq j$
            if $f(X_i^t)$ *is better than* $f(X_j^t)$ **then**
                $X_i^{t+1} = X_i^t + r(X^t{}_j - X^t{}_i)$
            end
            else
                $X_i^{t+1} = X_i^t + r(X^t{}_i - X^t{}_j)$
            end
            if $f(X_i^{t+1})$ *is better than* $f(X_i^t)$ **then**
                $X_i^t = X_i^{t+1}$
            end
        end
        Compute Q_m, I_m and D_m
        Fuzzify based on Q_m, I_m and D_m
        Defuzzify and set $Selection$ = crisp output
    end
    Get best result X_{best} from the current population X and put it in the final test suite list F_s
    Remove the interaction elements covered by X_{best} from the hash map H_s
end
Display F_s

Figure 3.13 ATLBO for Generating Mixed Strength t-way Test Suite

For further details, an illustration of the test suite generation process by ATLBO is shown in Figure 3.14.

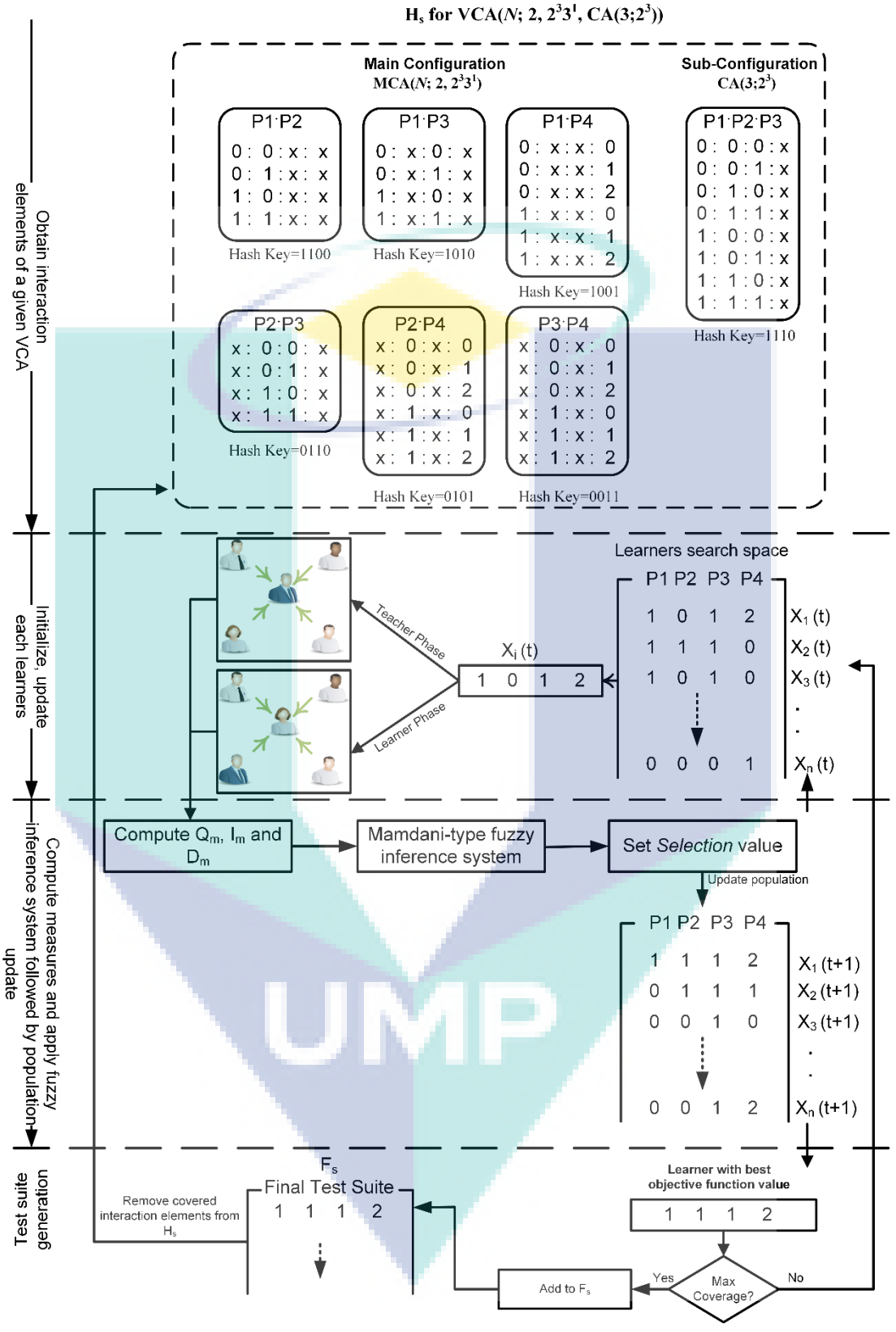


Figure 3.14 Graphical Representation of Test Suite Generation by ATLBO

Initially, ATLBO-based strategy receives H_s for a given CA or VCA. The strategy then randomly generates the search space as a population of learners. After this, the strategy activates either the teacher phase or the learner phase that updates each learner based on the maximum number of interaction elements coverage. For each of the learner in the population, ATLBO computes Q_m, I_m and D_m. Taking these measures as input linguistic variables, the Mamdani-type fuzzy inference system generates the crisp output *Selection*. This value is then used in the next run to decide whether to use the teacher phase or the learner phase. This procedure is repeated for all the learners in the population. Following this, the strategy selects the best test case from the updated population and check it for the maximum number of interaction elements coverage. If the test case covers more interaction elements, the strategy adds it to F_s. Finally, all the covered interaction elements are removed from H_s. Figure 3.15 illustrates an example of the generation of test suite and the removal of interaction elements or tuples from H_s.

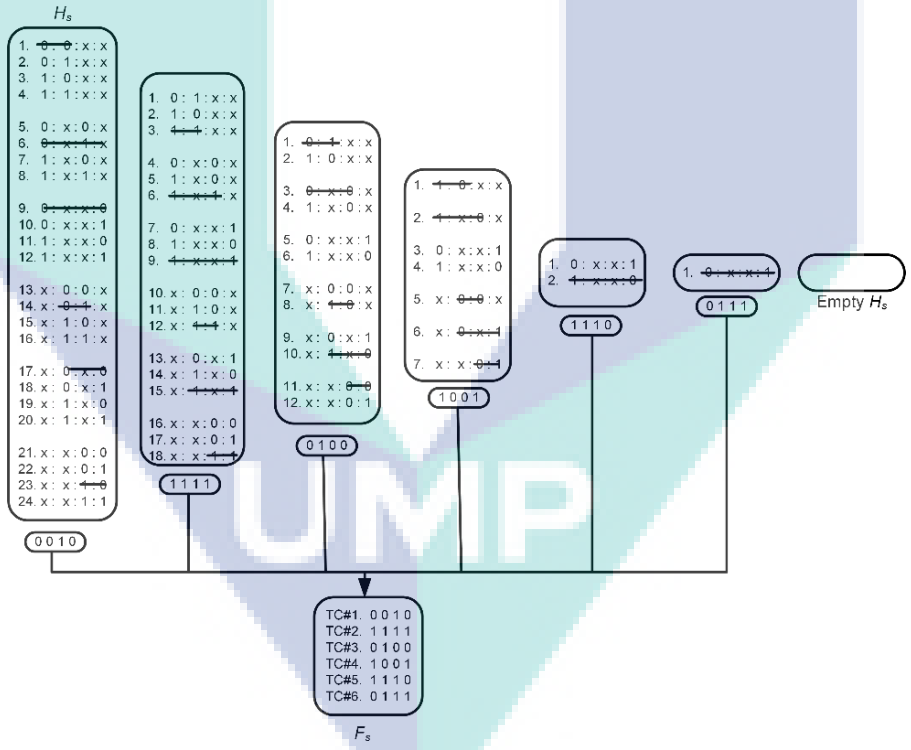


Figure 3.15 Example for Illustrating Generation of Test Suite and Removal of Interaction Elements from H_s

3.5 Chapter Summary

Summing up, the chapter has provided the complete technical details of this work by elaborating the design and implementation of all the algorithms. In the beginning, the

chapter has discussed TLBO which is a population-based optimization algorithm that mimics the classroom environment to produce optimal solutions. It has two main phases; the teacher phase and the learner phase, which are implemented sequentially. The teacher phase launches the global search operation, whereas the learner phase invokes the local search operation. Next, the chapter discusses the design of the proposed fuzzy inference system for ATLBO. The system has three inputs (Q_m, I_m and D_m) and one output (*Selection*). Trapezoidal member functions and four fuzzy inference rules have been used in the system. Afterward, the chapter has integrated the fuzzy inference system to present the general adaptive TLBO (ATLBO).

In the final part of the chapter, ATLBO is adopted for the problem of mixed strength *t*-way test suite generation. Here, the two main algorithms (i.e., interaction elements generation algorithm and ATLBO based test suite generator algorithm) are discussed in detail. The first algorithm initially generates the parameter (*P)* combinations and then the values *(v)* for each parameter combination based on interaction strength (*t*). Hash map approach, for storing and locating interaction elements, is preferred over the array list approach owing to its efficiency. The discrete version of ATLBO is used, and the clamming rule (absorbing walls) is established. ATLBO summons local and global search processes till all the interaction elements from H_s are completely removed (i.e., the stopping criteria for the processes). In the end, the chapter has illustrated how the ALBO automate the mixed strength test suite generation process. After design and implementation of ATLBO, the next chapter presents the experimental setup and detailed results to investigate and evaluate its performance and effectiveness against the original TLBO and other state-of-the-art meta-heuristic algorithms for addressing the problem of generating mixed strength test suite.

CHAPTER 4

RESULTS AND DISCUSSION

In the last chapter, design details of the general ATLBO, as well as its implementation for mixed strength test suite generation problem, have been provided. At the beginning, an overview of original TLBO has given. The Mamdani-type fuzzy inference system of ATLBO has been discussed in detail. ATLBO algorithm based on the fuzzy inference system has given. Finally, the implementation of ATLBO for the problem of generating mixed strength test suites has been given after explaining the interaction elements generation algorithm.

This chapter presents the evaluation process of ATLBO. For this purpose, the experiments are divided into three parts. The first part characterizes ATLBO and TLBO on a set of CAs and VCAs. The test suite generation time and sizes are used for the characterization. The second part adopts viable results from the literature for comparison with the results of both ATLBO and TLBO in terms of the generated test suite sizes. All other results are obtained by strategies based on meta-heuristic algorithms. The third part of the chapter presents the statistical analysis of the results obtained in the previous two parts. Moreover, the chapter also illustrates adaptive exploitation and exploration distribution pattern of ATLBO for each CA and VCA included in the experiments. Hereafter, the chapter thoroughly discusses the experimental observations. Finally, the chapter sheds light on the related threats to validities to the obtained experimental results.

4.1 Experimental Setup

For the effective evaluation of ATLBO, the experiments conducted in this study focuses on the following three related goals.

- To characterize the generation efficiency and performance of ATLBO against the original TLBO (i.e., the efficiency is characterized by the size of the generated test suite while the performance is characterized by the execution time of each strategy).

- To gauge the adaptive distribution pattern of the exploration and exploitation of ATLBO.

- To benchmark ATLBO against other meta-heuristic based test suite generation strategies including the original TLBO.

Both original TLBO and the proposed ATLBO are implemented using the same programming language and data structures as well as executed on the same hardware platform, their characterization is acceptable. Tracking the exploration and exploitation of ATLBO for all experiments is helpful to know how it carries search for different problems. The third goal ensures the performance evaluation of ATLBO against existing *t*-way strategies based on state-of-the-art meta-heuristic algorithms.

The experiments are divided into three parts to achieve the aforementioned goals. In the first part, 3 selected CAs: $CA(N; 2, 10^5)$, $CA(N; 2, 4^2\ 5^5)$, $CA(N; 2, 2^3\ 3^5)$, and 3 selected VCAs: $VCA(N; 2, 5^2\ 4^2\ 3^2, CA\ (3,4^2\ 3^2))$, $VCA(N; 2, 5^7, CA\ (3,5^3))$, $VCA(N; 2, 3^{13}, CA(3, 3^3))$ are adopted based on the interaction strength $t = 2$. In doing so, the aim is to highlight the time and size performance of the implemented ATLBO and the original TLBO. In the second part, the generated test suite sizes of the proposed ATLBO and the original TLBO implementations are benchmarked against each other as well as against existing meta-heuristic based strategies based on the benchmark experiments published in (Wu, Nie et al. 2015). To be specific, the benchmark experiments involve $CA(N; t, 3^p)$ with varying t from 2 to 4 and p from 2 to 12, $CA(N; t, v^7)$ along with $CA(N; t, v^{10})$ with varying t from 2 to 4, and v varying from 4 to 6, $VCA(N; 2, 3^{15}, \{C\})$, $VCA(N; 3, 3^{15}, \{C\})$, and $VCA(N; 2, 4^3\ 5^3\ 6^2, \{C\})$. In the third part, statistical analysis of the obtained results is carried out to further ensure an acceptable comparison.

It is noted that a fair comparison between each meta-heuristic based strategy (M. Črepinšek, S.-H. Liu et al. 2014, Draa 2015, M. Črepinšek, S.-H. Liu et al. 2015, M. Mernik, S.-H. Liu et al. 2015) is impossible owing to potentially different number of

fitness function evaluations, variation in the data structures, programming language implementation and running environment. Furthermore, each meta-heuristic may require the specific control parameter settings (e.g., PSO based strategies rely on inertia weight, social and cognitive parameters as parameters, whereas CS relies on its elitism probability). Given that the implementations of other meta-heuristic-based strategies are not available publicly, the algorithm internal settings cannot be modified so as to fairly run the adopted experiments in the experimental setup for this work. A *t*-way strategy based on the original TLBO for test suite generation is also implemented for comparative purposes. It is also observed that direct comparative performance of ATLBO with the original TLBO (i.e., even with the same number of iterations) can also be unfair. With the same number of iterations, the original TLBO has twice as much fitness function evaluations as compared to ATLBO owing to the serial execution of both exploration and exploitation steps. Thus, the number of iterations within TLBO must always be half of ATLBO for a fair comparison.

For ATLBO the population size is set to 40 and the maximum number of iterations is set to 100 in all the experiments. For the original TLBO, the population size is the same but the maximum number of iterations is set to 50. Both the ATLBO and TLBO implementations are based on the Java programming language. Table 4.1 presents the parameter settings for all the competing meta-heuristic algorithms. The experimental platform used for this work comprises of a PC running Windows 10, CPU 2.9 GHz Intel Core i5, 16 GB 1867 MHz DDR3 RAM and a 512 MB of flash HDD. ATLBO and TLBO are run 30 times in all the experiments to ensure statistical significance.

The best and the mean time (whenever applicable), as well as the best test sizes and the mean test sizes for each experiment, are reported side-by-side. The best cell entries are marked as "*", whereas the best mean cell entries are marked in bold font. Cell entries that are not available are marked with a dash "-". This study also tracks the mean percentage of exploration (i.e., global search) and exploitation (i.e., local search) for each experiment that involves ATLBO to highlight how the actual search progresses for different CAs and VCAs.

Table 4.1 Parameter Settings for the Competing Meta-heuristic Algorithms

Algorithm	Parameters	Values
SA	Starting temperature	20
	Cooling schedule	0.9998
	Iteration	1000
ACA	Pheromone control	1.6
	Heuristic control	0.2
	Pheromone amount	0.01
	Pheromone persistence	0.5
	Initial pheromone	0.4
	Elite ants	2
	Max stale period	5
	Population size	20
	Iteration	1000
PSTG (PSO)	Inertia weight	0.3
	Acceleration coefficients	1.375
	Population size	80
	Iteration	100
HS	Harmony memory consideration rate	0.7
	Pitch adjustment rate	0.2
	Harmony memory size Population size	100
	Iteration (improvisation)	100
CS	Probability	0.25
	Population size	100
	Iteration	100
DPSO	Inertia weight	0.5
	Acceleration coefficients	1.3
	Pro1	0.5
	Pro2	0.3
	Pro3	0.7
	Population size	80
	Iteration	250

4.2 Characterizing Time and Size Performances for TLBO and ATLBO

Given that both implementations are based on the same data structure, same language implementation, same running environment and same fitness function evaluations, a fair comparison of test suite sizes and time performance for TLBO and ATLBO is possible now. Table 4.2 highlights the obtained results, whereas Figure 4.1 depicts the mean exploration and exploitation percentage for ATLBO based on the provided CAs and VCAs. Moreover, box plots (see Figure 4.2) are constructed for each of the CA and VCA given in Table 4.2. These plots are helpful to visually show whether the implemented ATLBO and TLBO generate consistent results.

4.3 Benchmarking with other Meta-Heuristic Strategies

Unlike the experiments in the previous section, the benchmark experiments in this section also include the performance of ATBLO against all other strategies. However, the execution is omitted because of the differences in the parameter control settings (e.g., maximum iteration, unequal evaluation of fitness function, etc.) and implementation (e.g., data structure, implementation language, etc.). Despite these differences, it is the comparison is still valid as the published best and mean test sizes are obtained utilizing the best control parameter settings.

Tables 4.3-4.8 highlight the results for each CA and VCA considered in this study. Figures 4.3-4.8 depict the mean exploration and exploitation percentage for ATLBO based on the given CAs and VCAs.

4.4 Statistical Analysis

All the obtained results are analyzed statistically on the basis of 1xN pair comparisons with 95% confidence level (α=0.05) and 90% confidence level (α=0.10). The reason for using two values of α is that the competing strategies are not only well-tuned but also have reported their best-obtained test suite sizes. The Wilcoxon Rank-Sum test is adopted to prove the statistical significance of ATLBO-based strategy as the control strategy against other strategies in the comparison. The nature of the obtained results (being not normally distributed) justifies the selection of a non-parametric test such as the adopted Wilcoxon Rank-Sum test for the statistical analysis.

The null hypothesis H_0 demonstrates that the test suite size for the ATLBO-based strategy is statistically smaller than for the other competing strategy. In this case, ATLBO has a lower population median. The alternative hypothesis H_1 shows that there is no significant difference between ATLBO-based strategy and its counterpart in terms of test suite size.

The Bonferroni-Holm correction is adopted for the Family-Wise Error Rate (FWER) because of the multiple comparisons. It adjusts the α value by its step down procedure that sequentially rejects poor strategies than the control strategies until a better strategy arrives. To be specific, the α is adjusted by using Eq. 4.1 after sorting the *p-values* in ascending order such that $p_1 < p_2 < p_3 \ldots < p_i \ldots < p_k$.

$$\alpha_{Holm} = \frac{\alpha}{k-i+1} \quad 4.1$$

where k represents the total number of paired samples and i represents the current test number.

If $p_1 < \alpha_{Holm}$, the hypothesis for the comparison is rejected and a subsequent comparison for p_2 is then allowed. In case of rejection of the second hypothesis, the test proceeds to the third comparison. The procedure continues until a null hypothesis cannot be rejected. Tables 4.9-4.15 presents the complete statistical analyses.

4.5 Discussion

Reflecting on the experiments undertaken, several observations can be elaborated based on the obtained results.

Table 4.2 shows the size and time performance of ATLBO and the original TLBO. In terms of the best test size, ATLBO outperforms TLBO in two out of six entries. ATLBO outperforms TLBO in four out of six entries in terms of the mean test sizes. ATLBO and TLBO have similar execution times for small parameter values for time performances. However, TLBO significantly outperforms ATLBO as the parameter number increases (with fixed t = 2) because of the overhead introduced by the fuzzy inference selection. Figure 4.1 shows that the search gradually favours exploration over exploitation as parameter p increases (with constant t = 2, and a small variant of v).

The box plots analysis of Table 4.2 in Figure 4.2 (a)-(f) reveals some salient features of the searching process of ATLBO and TLBO. Considering Figure 4.2 (a) for *CA(N; 2, 10^5),* the distribution of box plot results is asymmetric for both strategies. TLBO has the largest range of results as well as has largest interquartile range than ATLBO. Moreover, the median of ATLBO is also lower than TLBO. Referring to Figure 4.2 (b) for *CA(N; 2, 4^2 5^5),* ATLBO has again the lower range of results than that of TLBO. However, both ATLBO and TLBO share the same interquartile range and median for the given CA. The distribution of box plot results for *CA(N; 2, 2^3 3^5)* shown in Figure 4.2 (c) is again asymmetric as both strategies have different ranges of results as well as different medians. ATLBO has a lowest range of results and median than TLBO. In this case, however, both have same interquartile range. Figure 4.2 (d) for *VCA(N; 5^2 4^2 3^2, CA(3, 5^2 4^2)* again demonstrates the dominance of ATLBO over TLBO as far as range of results,

median and interquartile range are concerned. Considering the box plots for the last two VCAs in Figure 4.2 (e) and Figure 4.2 (f), both ATLBO and TLBO have similar performances.

Unlike Table 4.2, Tables 4.3-4.8 account for the size performance of the proposed ATLBO and the original TLBO against other meta-heuristic-based strategies. The execution time measures are omitted in this case as the experiments are conducted unfairly based on unequal fitness function evaluation (e.g., different maximum number of iterations and control parameters).

ATLBO outperforms all other strategies with the best entries in 17 out of 24 cells in terms of the best test sizes as shown in Table 4.3. TLBO and DPSO also provide competitive performance with 14 and 12 best entries, respectively. APSO offers 7 best entries, whereas CS provides 5 best entries. PSTG performs the poorest with only 3 best entries. ATLBO also outperforms the rest of the strategies in terms of the mean test sizes (i.e., with 16 cells). The next closest rival is TLBO (i.e., with 5 cells) and DPSO (i.e., with 4 cells). From Figure 4.3 (a) till (c), it is observed that increasing the parameter value p for the same interaction strength t and values v causes ATLBO to favour exploration. Increasing t similarly also causes ATLBO to favour exploration.

DPSO outperforms other strategies with 12 out of 18 cell entries in terms of the best test sizes in Table 4.4. ATLBO comes as the runner up with total 8 best cells. Both APSO and CS offer 3 best cells, whereas TLBO offers 2 cells. PSTG again performs the poorest with only 1 best cell. DPSO also outperforms other strategies with 7 out of 18 best cell entries in terms of the mean test sizes. ATLBO is again the runner up with 6 best entries. APSO offers 2 best cell entries. TLBO, CS and PSTG perform the worst with only 1 best cell entry. Referring to Figure 4.4 (a) till (c), it is observed that increasing the values v for the same interaction strength t and parameters p have small effects in terms of exploration and exploitation. However, increasing t tends to cause ATLBO to increase exploration.

Table 4.2 Characterizing TLBO and ATLBO

ID	CA and VCA	Original TLBO				ATLBO				% Mean Exploit	%Mean Explore
		Size		Time (sec)		Size		Time (sec)			
		Best	Mean	Best	Mean	Best	Mean	Best	Mean		
CA1	CA(N; 2, 10^5)	117	118.60	28.92	42.30	116*	**118.40**	23.76*	**29.23**	79.85	20.16
CA2	CA(N; 2, $4^2 5^5$)	33	34.20	9.22*	**10.27**	32*	**33.90**	12.77	13.98	61.75	38.25
CA3	CA(N; 2, $2^3 3^5$)	13*	14.77	5.12*	**6.15**	13*	**14.16**	6.64	8.07	32.20	67.80
VCA1	VCA(N; 2, $5^2 4^2 3^2$, CA $(3,4^2 3^2)$)	105*	108.05	43.46*	**48.42**	105*	**107.60**	68.31	74.73	13.01	86.99
VCA2	VCA(N; 2, 5^7, CA $(3,5^3)$)	125*	**125.00**	66.63*	**69.10**	125*	**125.00**	125.02	131.42	18.69	81.31
VCA3	VCA(N; 2, 3^{13}, CA $(3,3^3)$)	27*	**27.00**	45.94*	**49.60**	27*	**27.00**	64.37	69.98	23.43	76.57

Entries with * indicate best sizes, entries in bold indicate best mean sizes

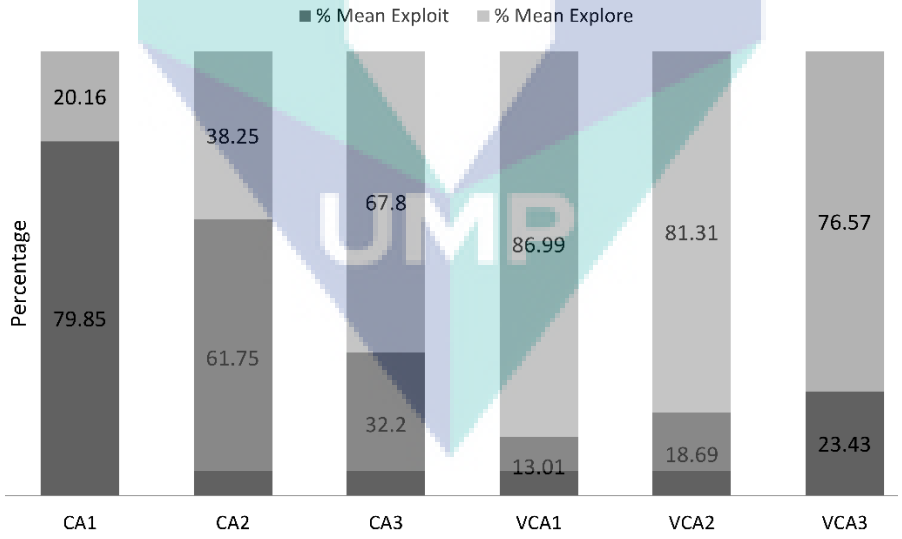


Figure 4.1 Mean Exploration and Exploitation Percentage of ATLBO for Table 4.2

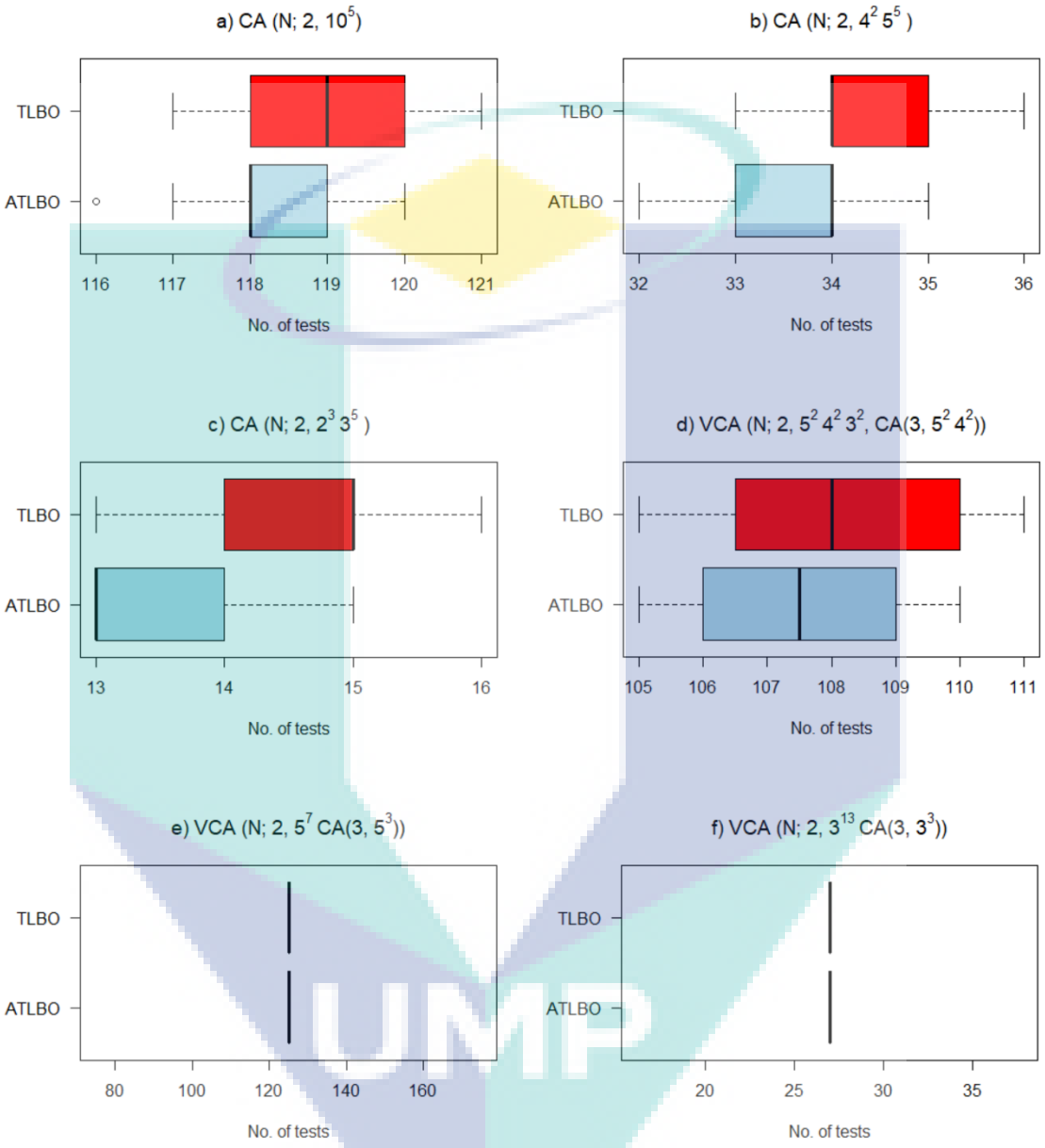


Figure 4.2 Box Plots for Table 4.2

Table 4.3 CA(N; t, 3^p)

t	*p*	PSTG (Ahmed, Zamli et al. 2012)		DPSO (Wu, Nie et al. 2015)		APSO (Mahmoud and Ahmed 2015)		CS (Ahmed, Abdulsamad et al. 2015)		Original TLBO		ATLBO			
		Best	Mean	Best	Mean	Best	Mean	Best	Mean	Best	Mean	Best	Mean	% Mean Exploit	% Mean Explore
2	4	9*	10.15	9*	**9.00**	9*	9.95	9*	10.0	9	**9.00**	9*	**9.00**	96.36	3.64
	5	12	13.81	11*	11.53	11*	12.23	11*	11.80	11*	11.43	11*	**11.33**	55.14	44.86
	6	13	15.11	14	14.50	12*	**13.78**	13	14.20	13	14.60	13	14.33	53.49	46.51
	7	15*	16.94	15*	15.17	15*	16.62	14*	15.60	15*	15.07	15*	**15.05**	52.71	47.29
	8	15*	17.57	15*	16.00	15*	16.92	15*	15.80	15*	**15.70**	15*	15.90	40.88	59.12
	9	17	19.38	15*	16.43	16	18.31	16	17.20	15*	16.23	15*	**15.03**	41.46	58.54
	10	17	19.78	16*	**17.30**	17	18.12	17	17.80	16*	17.40	16*	17.37	37.02	62.98
	11	17	20.16	17	17.70	-	-	18	18.60	16*	17.73	16*	**17.67**	36.77	63.23
	12	18	21.34	16*	17.93	-	-	18	18.80	17	18.10	17	**17.80**	37.14	62.86
3	5	39	41.37	41	43.17	41	42.20	38*	**39.20**	38*	42.53	38*	42.37	61.59	38.41
	6	45	46.76	33*	**38.30**	45	46.51	43	44.20	33*	38.87	33*	38.43	55.86	44.14
	7	50	52.20	48*	50.43	48*	51.12	48*	50.40	50	50.53	49	**50.00**	40.27	59.73
	8	54	56.76	52	53.83	50*	54.86	53	54.80	52	**53.17**	52	53.33	38.39	61.61
	9	58	60.30	56	57.77	59	60.21	58	59.80	56	57.77	55*	**57.50**	35.01	64.99
	10	62	63.95	59*	60.87	63	64.33	62	63.60	60	60.93	59*	**60.73**	34.09	65.91
	11	64	65.68	63	63.97	-	-	66	68.20	62*	63.70	62*	**63.57**	32.17	67.83
	12	67	68.23	65*	66.83	-	-	70	71.80	65*	66.70	65*	**66.53**	29.93	70.07
4	6	133	135.31	131	134.37	129*	133.98	132	134.20	130	**133.63**	130	134.10	50.50	49.50
	7	155	158.12	150	**155.23**	154	157.42	154	156.80	146*	155.77	152	156.03	40.22	59.78
	8	175	176.94	171*	175.60	178	179.70	173	174.80	171*	175.83	171*	**175.50**	33.85	66.15
	9	195	198.72	187	192.27	190	194.13	195	197.80	187	190.33	156*	**189.60**	31.76	68.24
	10	210	212.71	206	219.07	214	212.21	211	212.20	205*	208.80	207	**208.43**	27.20	72.80
	11	222	226.59	221	224.27	-	-	229	231.00	221*	**224.12**	221*	223.43	24.65	75.35
	12	244	248.97	237	239.83	-	-	253	255.80	236	239.29	235*	**237.83**	22.41	77.59

Entries with * indicate best sizes, entries in bold indicate best mean sizes and entries with '– 'indicate 'Not Available'

Table 4.4 CA(N; t, v^7)

t	*v*	PSTG (Ahmed, Zamli et al. 2012)		DPSO (Wu, Nie et al. 2015)		APSO (Mahmoud and Ahmed 2015)		CS (Ahmed, Abdulsamad et al. 2015)		Original TLBO		ATLBO			
		Best	Mean	Best	Mean	Best	Mean	Best	Mean	Best	Mean	Best	Mean	% Mean Exploit	% Mean Explore
2	2	6*	6.82	7	7.00	6*	**6.73**	6*	6.80	7	7.00	7	7.00	50.87	49.13
	3	15	15.23	14*	15.00	15	15.56	15	16.20	15	15.10	14*	**14.93**	51.60	48.40
	4	26	27.22	24	25.33	25	26.36	25	26.40	24	25.27	23*	**25.17**	57.74	42.26
	5	37	38.14	34*	35.47	35	37.92	37	38.60	34*	**35.43**	34*	35.47	63.82	36.18
	6	-	-	47*	49.23	-	-	-	-	47*	48.91	47*	**48.80**	70.29	29.71
	7	-	-	64*	66.37	-	-	-	-	65	66.21	64*	**66.10**	68.82	31.18
3	2	13	**13.61**	15	15.06	15	15.80	12*	13.80	15	15.12	15	15.12	48.42	51.58
	3	50	51.75	49	50.60	48*	51.12	49	51.60	49	50.38	48*	**50.33**	39.60	60.40
	4	116	118.13	112	**115.27**	118	120.41	117	118.40	112	115.37	111*	115.67	43.16	56.84
	5	225	227.21	216*	219.20	239	243.29	223	225.40	217	219.90	217	**218.80**	44.77	55.23
	6	-	-	365*	**370.57**	-	-	-	-	369	372.50	369	372.20	45.87	54.13
	7	-	-	574*	**577.67**	-	-	-	-	579	583.50	576	581.20	46.56	53.44
4	2	29	31.49	34	34.00	30	31.34	27*	**29.60**	31	33.70	31	33.68	46.04	53.96
	3	155	157.77	150*	**154.73**	153	155.20	155	156.80	151	155.25	150*	155.24	39.42	60.58
	4	487	489.91	472*	481.53	472*	**478.90**	487	490.20	480	485.53	478	484.69	39.90	60.10
	5	1176	1180.63	1148*	**1155.63**	1162	1169.94	1171	1175.20	1166	1173.17	1159	1173.40	40.14	59.86
	6	-	-	2341*	**2357.73**	-	-	-	-	2401	2406.35	2394	2404.25	40.47	59.53
	7	-	-	4290*	**4309.6**	-	-	-	-	4419	4421.40	4407	4417.60	37.38	62.62

Entries with * indicate best sizes, entries in bold indicate best mean sizes and entries with '– 'indicate 'Not Available'

Table 4.5 CA(N; t, v^{10})

t	*v*	PSTG (Ahmed, Zamli et al. 2012)		DPSO (Wu, Nie et al. 2015)		CS (Ahmed, Abdulsamad et al. 2015)		Original TLBO		ATLBO			
		Best	Mean	Best	Mean	Best	Mean	Best	Mean	Best	Mean	%Mean Exploit	%Mean Explore
2	4	-	-	28*	29.20	-	-	28*	28.73	28*	**28.69**	42.42	57.58
	5	45	48.31	42	43.67	45	47.8	41*	**43.30**	42	43.53	46.92	53.08
	6	-	-	58*	**59.23**	-	-	58*	59.47	58*	59.33	50.27	49.73
3	4	-	-	141	143.70	-	-	140*	**142.57**	140*	142.80	30.77	69.23
	5	287	298.00	273	276.20	297	299.20	273	275.70	272*	**275.23**	31.04	68.96
	6	-	-	467	470.50	-	-	467	470.47	466*	**469.90**	31.53	68.47
4	4	-	-	664	667.00	-	-	663	668.12	661*	**664.06**	25.68	74.32
	5	1716	1726.72	1618*	**1620.80**	1731	1740.20	1621	1621.80	1619	1620.91	22.32	77.68
	6	-	-	3339	3342.50	-	-	3338*	3343.81	3338*	**3342.10**	21.13	78.87

Entries with * indicate best sizes, entries in bold indicate best mean sizes and entries with '– 'indicate 'Not Available'

Table 4.6 VCA(N; 2, 3^{15}, {C})

ID	VCA	PSTG (Ahmed, Zamli et al. 2012)		DPSO (Wu, Nie et al. 2015)		ACS (Shiba, Tsuchiya et al. 2004)		SA (Cohen, Colbourn et al. 2003)		Original TLBO		ATLBO			
		Best	Mean	Best	Mean	Best	Mean	Best	Mean	Best	Mean	Best	Mean	%Mean Exploit	%Mean Explore
VCA1	Ø	19	20.92	18	18.63	19	-	16*	-	19	19.67	18	**18.60**	31.30	68.70
VCA2	CA(3, 3^3)	27*	27.50	27*	27.27	27*	-	27*	-	27*	27.33	27*	**27.00**	22.26	77.74
VCA3	CA(3, 3^3)^2	27*	27.94	27*	27.83	27*	-	27*	-	27*	**27.47**	27*	27.53	21.46	78.54
VCA4	CA(3, 3^3)^3	27*	28.13	27*	28.00	27*	-	27*	-	27*	27.93	27*	**27.43**	22.00	78.00
VCA5	CA(3, 3^4)	30	31.47	27*	31.43	27*	-	27*	-	27*	32.73	27*	**27.00**	22.26	77.74
VCA6	CA(3, 3^5)	38	**39.83**	38	40.93	38	-	33*	-	38	40.97	38	40.60	16.25	83.75
VCA7	CA(3, 3^6)	45	46.42	43	45.70	45	-	34*	-	43	43.73	43	**43.67**	18.05	81.95
VCA8	CA(3, 3^7)	49	51.68	47	49.87	48	-	41*	-	49	50.03	47	**49.83**	17.96	82.04
VCA9	CA(4, 3^4)	81*	82.21	81*	**81.03**	-	-	-	-	81*	**81.03**	81*	**81.03**	7.44	92.56
VCA10	CA(4, 3^5)	97	99.31	85*	**94.50**	-	-	-	-	89	97.53	87	96.90	7.26	92.74
VCA11	CA(4, 3^7)	158	160.31	152*	156.83	-	-	-	-	153	156.51	152*	**156.33**	10.74	89.26

Entries with * indicate best sizes, entries in bold indicate best mean sizes and entries with '– 'indicate 'Not Available'

Table 4.7 VCA(N; 3, 3^{15}, {C})

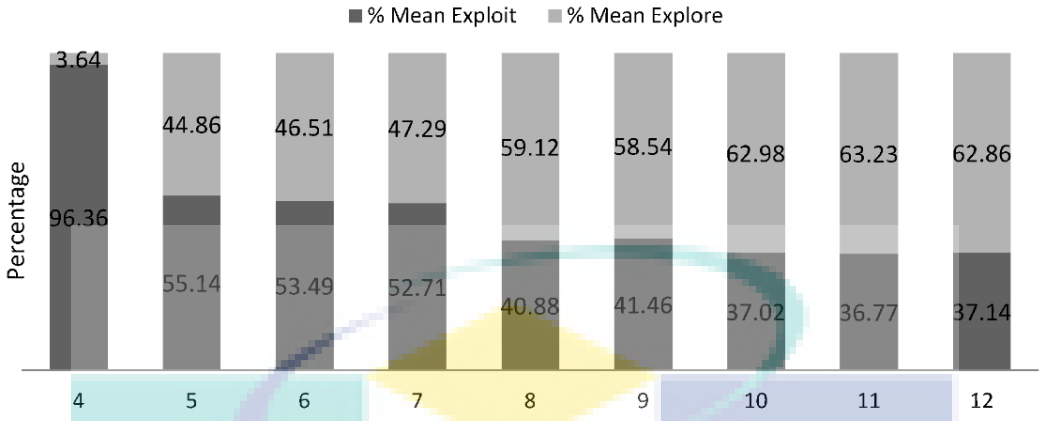
ID	VCA	PSTG (Ahmed, Zamli et al. 2012)		DPSO (Wu, Nie et al. 2015)		HSS (Alsewari and Zamli 2012)		Original TLBO		ATLBO			
		Best	Mean	Best	Mean	Best	Mean	Best	Mean	Best	Mean	%Mean Exploit	%Mean Explore
VCA1	Ø	75	78.69	72*	73.97	75	75.00	73	74.47	73	**73.60**	24.64	75.36
VCA2	CA(4, 3^4)	91	91.80	86	89.83	87	**87.00**	90	90.03	85*	89.23	20.36	79.64
VCA3	CA(4, 3^4)^2	91	92.21	88	90.77	90	90.00	86*	**89.76**	87	90.10	20.24	79.76
VCA4	CA(4, 3^5)	114	117.30	107	**111.17**	112	112.00	106*	111.90	107	112.13	16.44	83.56
VCA5	CA(4, 3^7)	159	162.23	152*	158.57	159	160.10	155	158.40	153	**158.30**	12.11	87.89
VCA6	CA(4, 3^9)	195	199.28	193	196.00	199	199.80	190	193.40	189*	**193.29**	11.15	88.85
VCA7	CA(4, 3^{11})	226	230.64	225*	227.50	242	243.00	226	229.51	225*	**227.48**	10.01	89.99

Entries with * indicate best sizes, entries in bold indicate best mean sizes and entries with '– 'indicate 'Not Available'

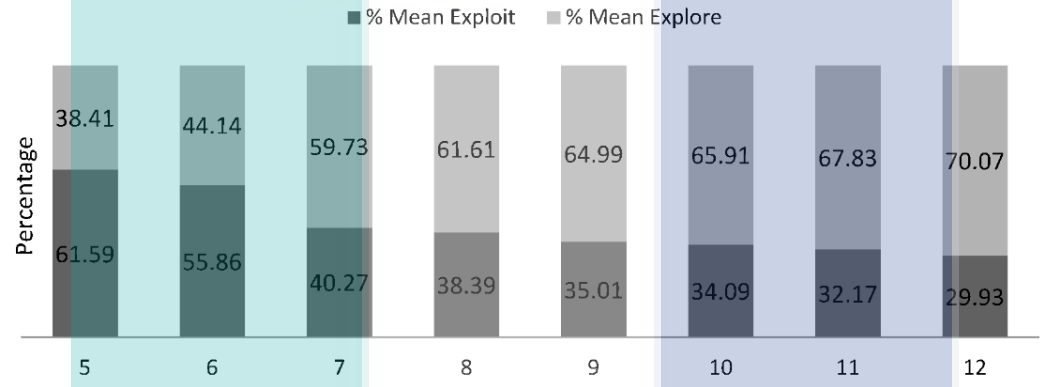
Table 4.8 VCA(N; 2, 4^3 5^3 6^2, {C})

ID	CA	PSTG (Ahmed, Zamli et al. 2012)		DPSO (Wu, Nie et al. 2015)		HSS (Alsewari and Zamli 2012)		ACS (Shiba, Tsuchiya et al. 2004)		SA (Cohen, Colbourn et al. 2003)		Original TLBO		ATLBO			
		Best	Mean	Best	Mean	Best	Mean	Best	Mean	Best	Mean	Best	Mean	Best	Mean	%Mean Exploit	%Mean Explore
VCA1	Ø	42	43.60	40	42.30	42	43.50	41	-	36*	-	40	42.03	39	41.63	43.47	56.53
VCA2	CA(3, 4^3)	64*	65.50	64*	**64.00**	64*	**64.00**	64*	-	64*	-	64*	64.03	64*	**64.00**	31.11	68.89
VCA3	CA(3, 4^3 5^2)	124	126.60	119	124.70	116	**120.90**	104	-	100*	-	121	125.67	122	124.50	18.31	81.69
VCA4	CA(3, 4^3), CA(3, 5^3)	125*	127.90	125*	**125.00**	125*	**125.00**	125*	-	125*	-	125*	**125.00**	125*	**125.00**	15.88	84.12
VCA5	CA(3, 4^3 5^3 6^1)	206	210.20	203	**207.50**	212	214.00	201	-	171*	-	203	208.77	203	208.68	14.42	85.58
VCA6	CA(3, 4^3), CA(4,5^3 6^1)	750*	755.70	750*	750.80	750*	**750.00**	-	-	-	-	750*	**750.00**	750*	**750.00**	12.70	87.30
VCA7	CA(4, 4^3 5^2)	472	478.10	440*	**450.60**	453	454.3	-	-	-	-	459	466.70	451	459.10	6.52	93.48

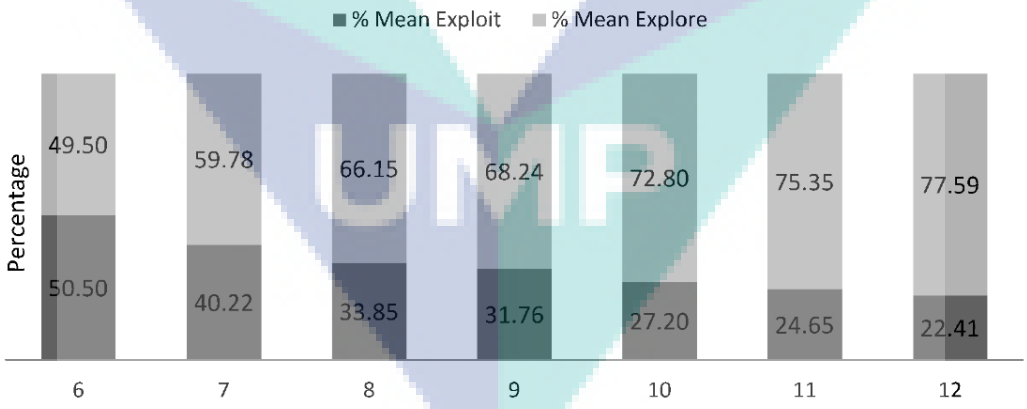
Entries with * indicate best sizes, entries in bold indicate best mean sizes and entries with '– 'indicate 'Not Available'



a) CA (N; t, 3^p) with $t = 2$, p is varied from 4 to 12

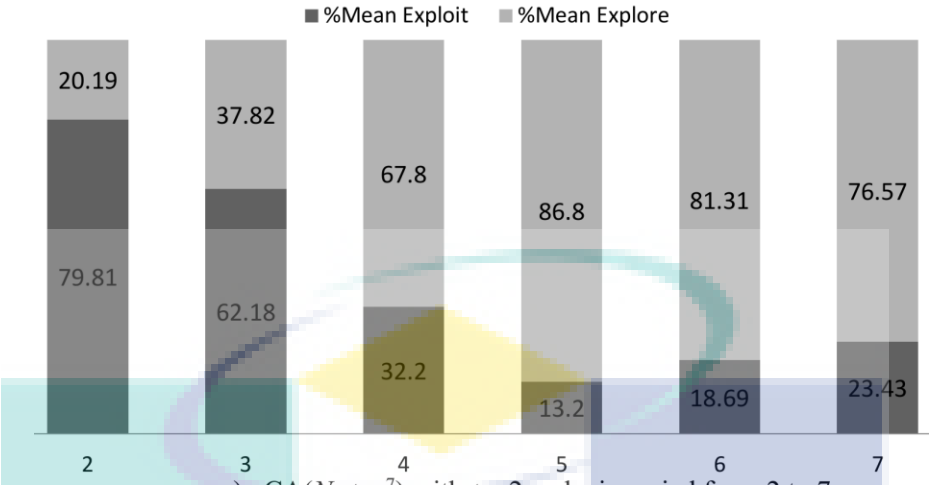


b) CA (N; t, 3^p) with $t = 3$, p is varied from 5 to 12

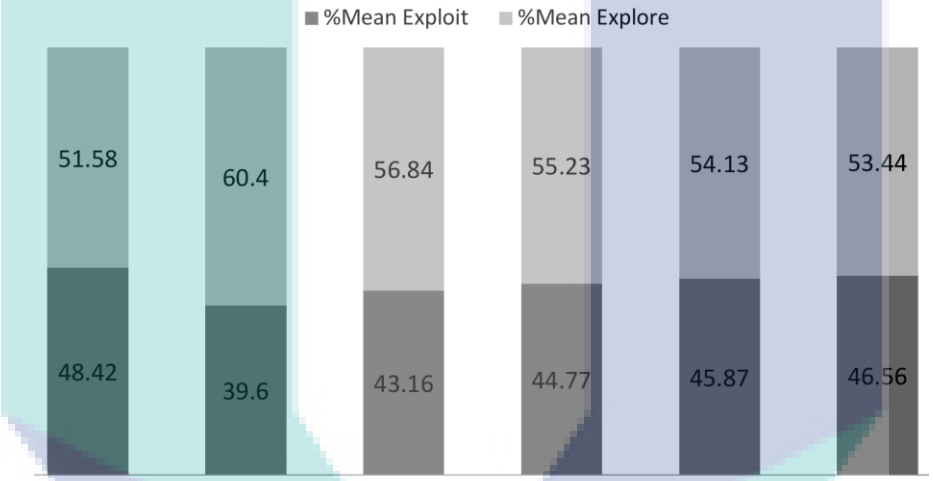


c) CA (N; t, 3^p) with $t = 4$, p is varied from 6 to 12

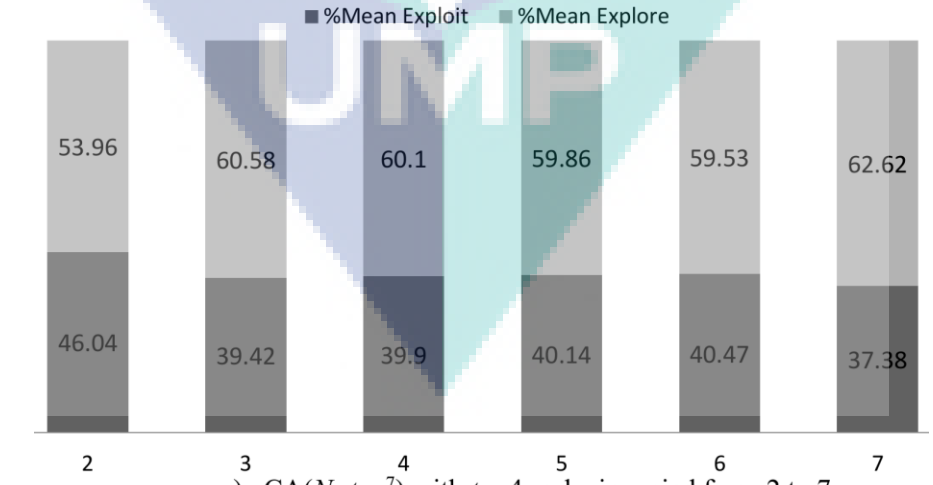
Figure 4.3 Mean Exploration and Exploitation Percentage of ATLBO for Table 4.3



a) CA(N; t, v^7) with $t = 2$ and v is varied from 2 to 7



b) CA(N; t, v^7) with $t = 3$ and v is varied from 2 to 7



c) CA(N; t, v^7) with $t = 4$ and v is varied from 2 to 7

Figure 4.4 Mean Exploration and Exploitation Percentage of ATLBO for Table 4.4

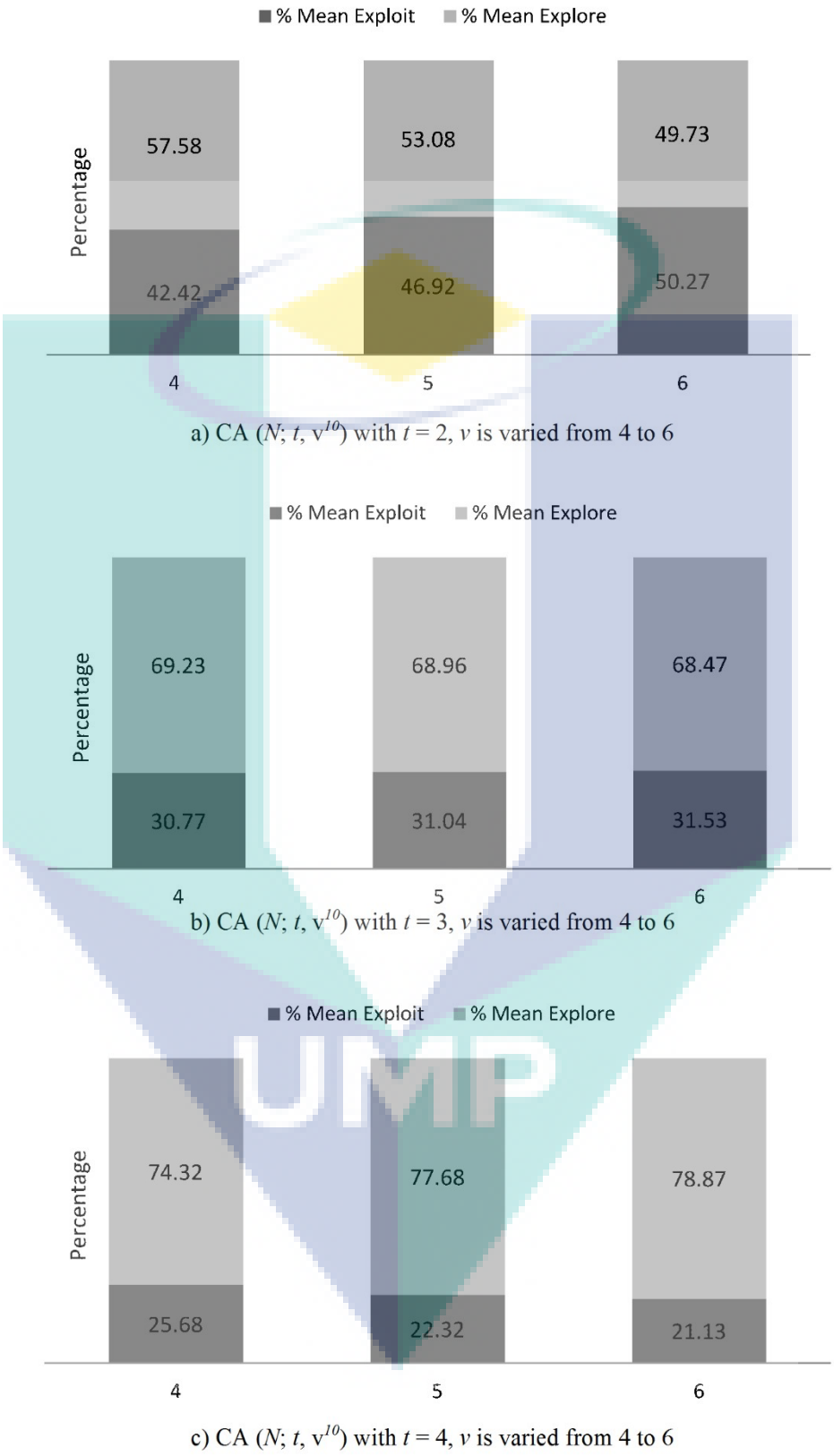


Figure 4.5 Mean Exploration and Exploitation Percentage of ATLBO for Table 4.5

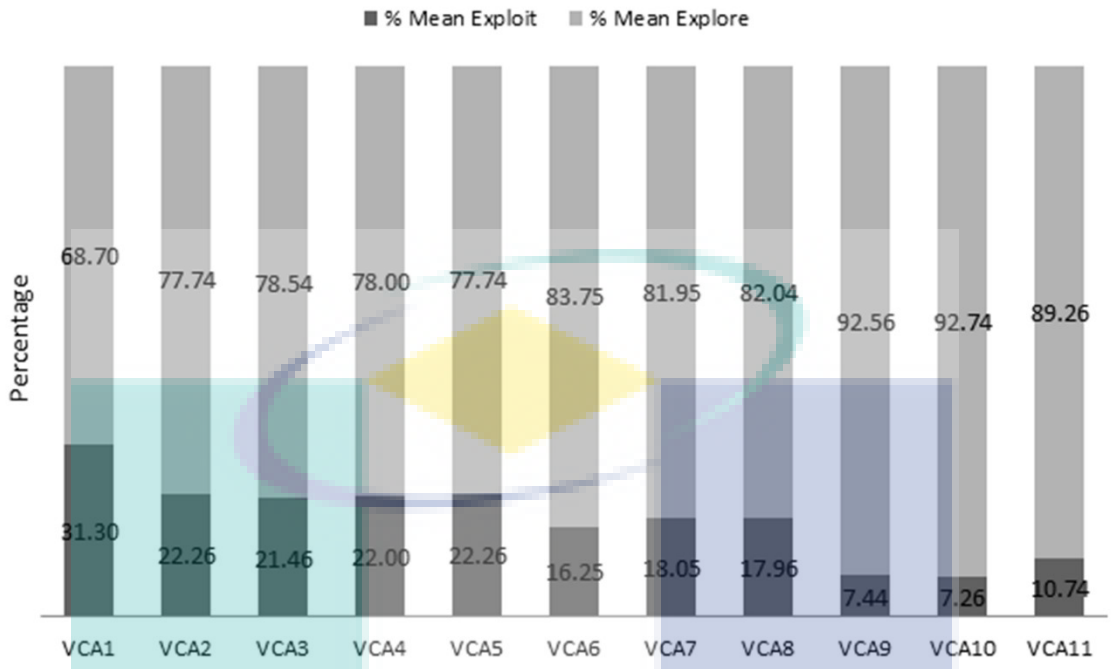


Figure 4.6 Mean Exploration and Exploitation Percentage of ATLBO for Table 4.6

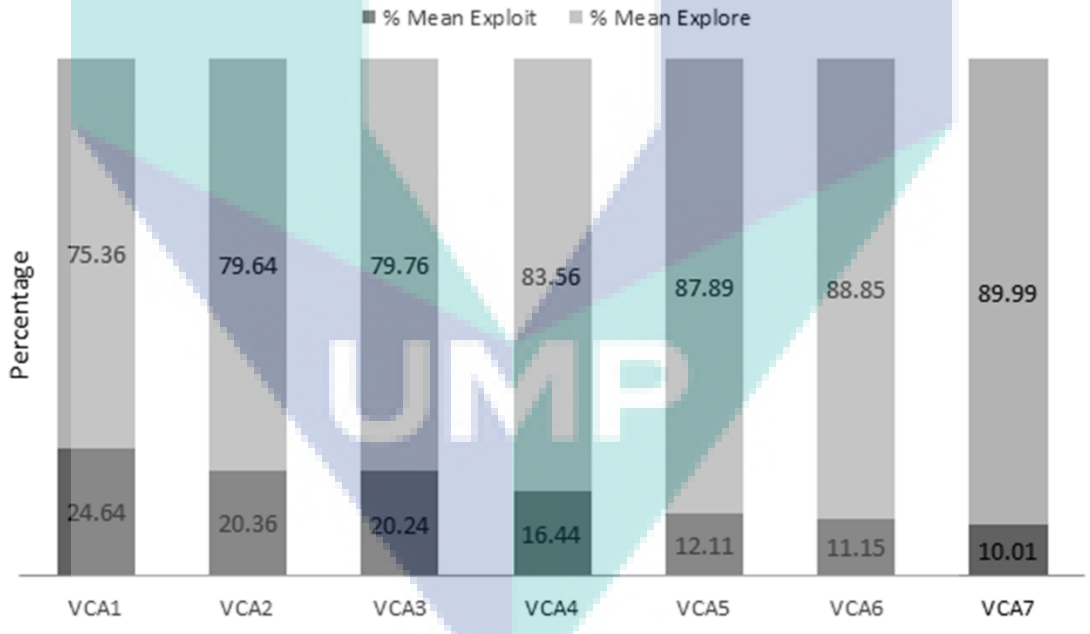


Figure 4.7 Mean Exploration and Exploitation Percentage of ATLBO for Table 4.7

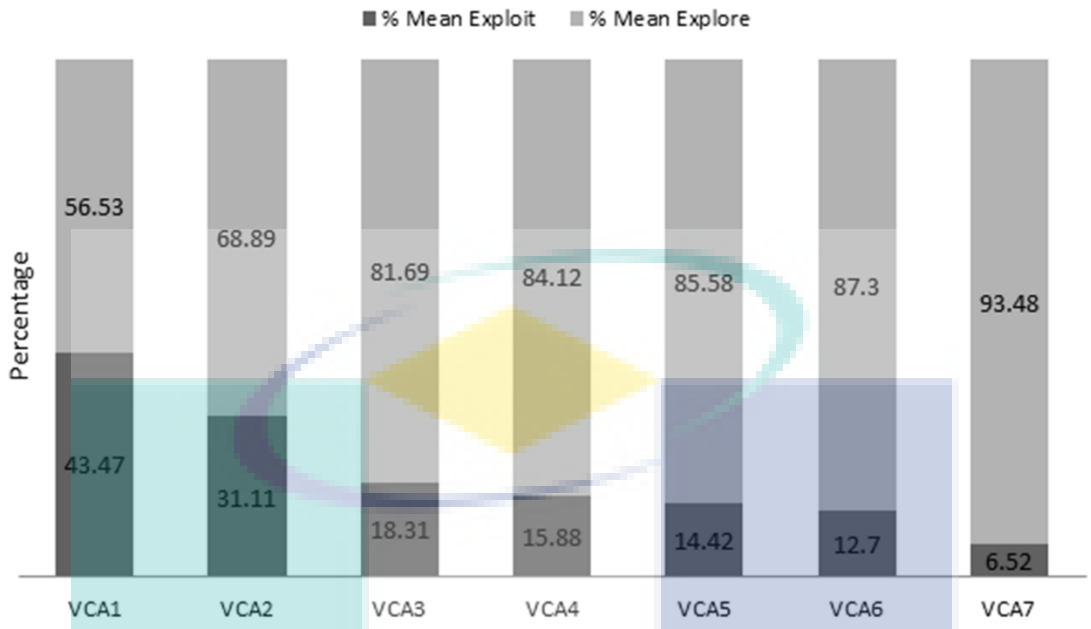


Figure 4.8 Mean Exploration and Exploitation Percentage of ATLBO for Table 4.8

Table 4.9 Wilcoxon Rank-Sum Test for Table 4.2

Pair Comparison	p-value in ascending order	Bonferroni-Holm Correction p_{holm} (95% confidence level)	Bonferroni-Holm Correction p_{holm} (90% confidence level)
ATLBO vs TLBO	0.0215	$p_{holm} = 0.05$, p-value < p_{holm},Reject H_o	$p_{holm} = 0.10$, p-value < p_{holm}, Reject H_o

Table 4.10 Wilcoxon Rank-Sum Test for Table 4.3

Pair Comparison	p-value in ascending order	Bonferroni-Holm Correction p_{holm} (95% confidence level)	Bonferroni-Holm Correction p_{holm} (90% confidence level)
ATLBO vs PSTG	0.0000	$p_{holm} = 0.0125$, p-value < p_{holm}, Reject H_o	$p_{holm} = 0.025$, p-value < p_{holm}, Reject H_o
ATLBO vs CS	0.0005	$p_{holm} = 0.016667$, p-value < p_{holm}, Reject H_o	$p_{holm} = 0.$ 0.033333, p-value < p_{holm}, Reject H_o
ATLBO vs DPSO	0.0005	$p_{holm} = 0.025$, p-value < p_{holm}, Reject H_o	$p_{holm} = 0.05$, p-value < p_{holm}, Reject H_o
ATLBO vs TLBO	0.0025	$p_{holm} = 0.05$, p-value < p_{holm}, Reject H_o	$p_{holm} = 0.10$, p-value < p_{holm}, Reject H_o

Owing to incomplete sample (i.e., with one or more NA entries), the contribution APSO are ignored

Table 4.11 Wilcoxon Rank-Sum Test for Table 4.4

Pair Comparison	p-value in ascending order	Bonferroni-Holm Correction p_{holm} (95% confidence level)	Bonferroni-Holm Correction p_{holm} (90% confidence level)
ATLBO vs TLBO	0.017	$p_{holm} = 0.025$, p-value < p_{holm}, Reject H_o	$p_{holm} = 0.02$, p-value < p_{holm}, Reject H_o
ATLBO vs DPSO	0.1025	$p_{holm} = 0.05$, p-value > p_{holm}, Cannot Reject H_o	$p_{holm} = 0.10$, p-value > p_{holm}, Cannot Reject H_o

Owing to incomplete sample (i.e., with one or more NA entries), the contributions of PSTG, CS and APSO are ignored

Table 4.12 Wilcoxon Rank-Sum Test for Table 4.5

Pair comparison	p-value in ascending order	Bonferroni-Holm correction p_{holm} (95% confidence level)	Bonferroni-Holm correction p_{holm} (90% confidence level)
ATLBO vs DPSO	0.0105	$p_{holm} = 0.025$, p-value < p_{holm}, Reject H_o	$p_{holm} = 0.05$, p-value < p_{holm}, Reject H_o
ATLBO vs TLBO	0.033	$p_{holm} = 0.05$, p-value < p_{holm}, Reject H_o	$p_{holm} = 0.10$, p-value < p_{holm}, Reject H_o

Owing to incomplete sample (i.e., with one or more NA entries), the contributions of PSTG and CS are ignored

Table 4.13 Wilcoxon Rank-Sum Test for Table 4.6

Pair comparison	p-value in ascending order	Bonferroni-Holm correction p_{holm} (95% confidence level)	Bonferroni-Holm correction p_{holm} (90% confidence level)
ATLBO vs TLBO	0.004	$p_{holm} = 0.016667$, p-value < p_{holm}, Reject H_o	$p_{holm} = 0.033333$, p-value < p_{holm}, Reject H_o
ATLBO vs PSTG	0.005	$p_{holm} = 0.025$, p-value < p_{holm}, Reject H_o	$p_{holm} = 0.05$ p-value < p_{holm}, Reject H_o
ATLBO vs DPSO	0.0295	$p_{holm} = 0.05$, p-value < p_{holm}, Reject H_o	$p_{holm} = 0.10$, p-value < p_{holm}, Reject H_o

Owing to incomplete sample (i.e., with one or more NA entries), the contributions of ACS and SA are ignored

Table 4.14 Wilcoxon Rank-Sum Test for Table 4.7

Pair comparison	p-value in ascending order	Bonferroni-Holm correction p_{holm} (95% confidence level)	Bonferroni-Holm correction p_{holm} (90% confidence level)
ATLBO vs PSTG	0.009	p_{holm} = 0.0125, p-value < p_{holm}, Reject H_o	p_{holm} = 0.025, p-value < p_{holm}, Reject H_o
ATLBO vs HSS	0.009	p_{holm} = 0.016667, p-value < p_{holm}, Reject H_o	p_{holm} = 0.033333, p-value < p_{holm}, Reject H_o
ATLBO vs TLBO	0.118	p_{holm} = 0.025 , p-value > p_{holm}, Cannot Reject H_o	p_{holm} = 0.05 p-value > p_{holm}, Cannot Reject H_o
ATLBO vs DPSO	0.199	p_{holm} = 0.05 , p-value > p_{holm}, Cannot Reject H_o	p_{holm} = 0.10, p-value > p_{holm}, Cannot Reject H_o

Table 4.15 Wilcoxon Rank-Sum Test for Table 4.8

Pair comparison	p-value in ascending order	Bonferroni-Holm correction p_{holm} (95% confidence level)	Bonferroni-Holm correction p_{holm} (90% confidence level)
ATLBO vs PSTG	0.009	p_{holm} = 0.0125, p-value < p_{holm}, Reject H_o	p_{holm} = 0.025, p-value < p_{holm}, Reject H_o
ATLBO vs TLBO	0.0215	p_{holm} = 0.016667, p-value > p_{holm}, Cannot Reject H_o	p_{holm} = 0.033333, p-value < p_{holm}, Reject H_o
ATLBO vs DPSO	0.343	p_{holm} = 0.025 , p-value > p_{holm}, Cannot Reject H_o	p_{holm} = 0.05 p-value > p_{holm}, Cannot Reject H_o
ATLBO vs HSS	0.50	p_{holm} = 0.05 , p-value > p_{holm}, Cannot Reject H_o	p_{holm} = 0.10, p-value > p_{holm}, Cannot Reject H_o

Owing to incomplete sample (i.e., with one or more NA entries), the contributions of ACS and SA are ignored

Given the lack of published results, few observations can be made for PSTG and CS in Table 4.5. ATLBO offers the best results in almost all configurations in terms of the best test sizes with the exception of $CA(N; 2, 5^{10})$ and $CA(N; 4, 5^{10})$. As the runner up, TLBO (i.e., 5 out of 9 best cell entries) outperforms DPSO (i.e., 3 out of 9 best cell entries). ATLBO outperforms both DPSO and TLBO with 5 out 9 best entries in terms of the mean test sizes. Both DPSO and TLBO share the same number of best mean test sizes (i.e., 2 out of 9 best cell entries). The exploration and exploitation of ATLBO in Figure 4.5 (a) till (c), show that increasing the values v for the same interaction strength t and parameters p causes a small increase in exploitation. Increasing t tends to cause ATLBO to increase exploration, which is similar to an earlier case.

Table 4.6 indicates that SA outperforms all other strategies in terms of the best test sizes with 8 out of 11 best cell entries. DPSO is the runner up with 7 best cell entries. Except SA and DPSO, ATLBO outperforms all other strategies with 6 best cells. TLBO outperforms PSTG and ACS with 5 best cells. PSTG and ACS perform the worst with 4

cells. ATLBO outperforms all other strategies with 8 out of 11 best cells in terms of the mean test sizes. Although DPSO has the best results, it has a poorer mean value compared with ATLBO with 3 out of 11 cells. TLBO offers 2 cells with the best mean test sizes. PSTG has one entry with the best mean value. Given the lack of published results, information cannot be inferred for ACS and SA. Figure 4.6 shows that increasing sub-configurations causes ATLBO to further increase exploration than exploitation for the fixed *VCA(N; 2, 3^{15}, {C})* with sub-configurations *{C}*.

Both DPSO and ATLBO outperform all other strategies in Table 4.7 in terms of the best test sizes with 3 out of 7 cell entries, followed by TLBO with 2 cells. HSS and PSTG perform the worst without a single best cell entry. ATLBO outperforms all existing strategies with 4 out 7 entries in terms of the mean test sizes. TLBO, HSS and DPSO offer only 1 best cell entry. PSTG performs the poorest with no best mean value. The chart in Figure 4.7 shows that increasing sub-configurations also tend to increase exploration further for the fixed *VCA(N; 2, 3^{15}, {C})* with sub-configurations *{C}*.

SA outperforms all other strategies in all VCA configurations in terms of the best test sizes with 5 entries, as shown in Table 4.8. DPSO follows with 4 best cell entries, which performs better than ATLBO, TLB, HSS and PSTG (all with three cell entries, respectively). ACS has the poorest performance with 2 cell entries. DPSO and HSS yield the best results with 4 out of 7 entries for the mean test sizes. ATLBO is the runner up with 3 cell entries. TLBO has 2 best entries, whereas PSTG has none. Information cannot be inferred for ACS and SA because of the lack of published results. Finally, the chart in Figure 4.8 shows consistent observation noted for the last two findings. In particular, increasing sub-configurations *{C}* also tend to increase exploration further for the fixed *VCA(N; 2, 4^3 5^3 6^2, {C})*.

Considering both comparisons (best and mean test suite sizes), ATLBO offers competitive performance against existing strategies (with the closest competitors are DPSO and SA). Despite its competitive performance, there appears to be some overhead for ATLBO as far as execution time is concerned, that is, in order to accommodate the processes related to fuzzy inference rules (i.e., calculating the quality measure (Q_m), intensification measure (I_m), and diversification measure (D_m)).

Recall that this work modifies the sequential nature of exploration and exploitation within TLBO. Specifically, this work enhances the original TLBO with the adaptive selection – local search and global search are decided at run-time based on the progress of the search. In fact, the core feature of TLBO has been maintained, that is, the proposed ATLBO is still *parameter-free*. Typically, existing meta-heuristics adopts specific control parameters and requires explicit (problem domain) tuning to ensure a balance between exploration and exploitation. Explicit tuning is unnecessary for ATLBO because the balance between exploration and exploitation is adaptively handled by the implemented fuzzy inference system.

The search pattern of the original TLBO is straightforward, wherein both exploration and exploitation are always at 50%. However, to understand the searching pattern of ATLBO, there is a need to track the mean percentage of exploration and exploitation taking the mixed strength t-way test generation problem as the case study. Within the problem of generating mixed t-way test suites, $VCA(N; t, v^p, \{C\})$, four main variables of interest exist (i.e., interaction strength t, values v, parameter p, and sub-configuration $\{C\}$). Based on the conducted experiments, the following conclusions have been derived.

- For small value of p, t and v, ATLBO favours exploitation over exploration (i.e., with typical values $p \leq 6$, $t \leq 3$, $v \leq 2$).
- For a fixed t and v, when the parameter p increases, ATLBO favours exploration over exploitation.
- For a fixed p and v, when the interaction strength t increases, ATLBO favours exploration over exploitation.
- For a fixed p and t, when the value v increases, ATLBO favours exploration over exploitation. However, the rate of exploration increment is smaller as compared to the effect of increasing p or t.
- Given $VCA(N; t, v^p, \{C\})$, and for a fixed p, v, and t, when $\{C\}$ increases, ATLBO favours exploration over exploitation.

As the search space grows (i.e. horizontally with the increase of p and t, or vertically with the increase of v), ATLBO needs to explore more promising regions to obtain good quality solutions. The findings of this work are consistent with intuition indicating the effectiveness of the developed fuzzy rules.

ATLBO-based strategy demonstrated acceptable statistical dominance against the competing state-of-the-art strategies as shown in Tables 4.9-4.15 by the conducted statistical tests for all the obtained test sizes. Statistical analysis of Table 4.2 presented in Table 4.9 clearly indicates statistical significance of ATLBO-based strategy in performance against the original TLBO at both values of α (i.e., 95% confidence level and 90% confidence level).

According to the statistical analysis of Table 4.3 in Table 4.10, ATLBO is statistically significant than PSTG, CS, DPSO and TLBO while ignoring the contribution of APSO owing to missing results. Table 4.11 shows the statistical analysis of Table 4.4, indicating that statistically, ATLBO has better performance than TLBO-based strategy, whereas similar performance to DPSO-based strategy. Here, the contributions of PSTG, CS and APSO are ignored due to the unavailability of complete samples for these strategies. Referring to the statistical analysis of Table 4.5 in Table 4.12, ATLBO is statistically better than DPSO and TLBO at both 95% confidence level and 90% confidence level, whereas the contributions of PSTG and CS are excluded because of the incomplete samples. At both 95% confidence level and 90% confidence level, ATLBO-based strategy has statistically significant performance than PSTG, TLBO and DPSO according to statistical analysis of Table 4.6 in Table 4.13. ATLBO has outperformed PSTG and HSS statistically at both 95% confidence level and 90% confidence level, whereas showed similar performance against TLBO and DPSO (see statistical analysis of Table 4.7 in Table 4.14). Statistical analysis of Table 4.8 in Table 4.15 indicates that statistically ATLBO has better performance than PSTG and TLBO at 90% confidence level, whereas it has similar performance to other strategies at both 95% and 90% confidence levels.

4.6 Threats to Validity

Many threats to validities could be associated with the experimental studies. Few threats have been identified in this research and subsequently elaborated so as to mitigate their effects on the obtained results.

First, the choice of the benchmarks represents an essential threat. The experimental benchmarks adopted in this work are from other well-known studies and experiments in the literature. However, it cannot be guaranteed that these benchmarks represent the actual software configurations in real world. Nevertheless, the benchmarks are derived from configurations of different software programs.

Second, a comparison with other strategies is another threat. Many strategies and tools for generating the *t*-way test suite exist. Given the unavailability of these strategies for implementation within the experimental environment set up for this work, ALTBO cannot be compared with all the available strategies. To eliminate this threat, recently published results in a reputable journal for those highest related strategies which are closed to ALTBO (e.g., (Wu, Nie et al. 2015)) have chosen. In the case of this work, the tuning of the parameters of those strategies is out of the control. Nevertheless, the comparison here is valid because the published results were obtained with the best tuning parameters.

Third, the number of fitness function evaluations can be considered a threat. The original TLBO has twice as much fitness function evaluations as ATLBO, which can also be a significant threat to the experimentations, rendering unfair comparisons. Both the teacher and student phase processes are serially executed per iteration in the original TLBO. In contrast, only one process is selected per iteration in ATLBO based on the adaptive measure of the searching process. Given that both of the implementations have been done for this research, straightforwardly these threats can be eliminated. To be specific, it can be ensured that the number of iterations within TLBO is always half of that of the ATLBO.

Fourth, the randomness of the search operators within the meta-heuristic strategies can also be an issue. Here, the best test size results can potentially be obtained at this point by chance and only once out of many runs. Reporting and comparing only the best test size results might not provide a fair indication of the size performance of a

particular strategy. Thus, this work also relied on the mean results rather than merely focusing on the best test size results.

Fifth, the choice of fuzzy implementation is another important threat. Using different fuzzy implementation and inference system may lead to different results (with different membership functions). It is recognized that at least two different fuzzy inference systems variations exist in the literature (e.g., the Mamdani-type versus Sugeno-type fuzzy inference systems). Previous studies that adopt fuzzy systems to control various parameters within meta-heuristic algorithm apply the Mamdani-type inference utilizing the centre of gravity for the output defuzzification (Mamdani and Assilian 1975). In fact, most studies often employ either trapezoidal (i.e., a variant of triangular) or Gaussian membership functions. In one such study by Valdez et al. (Valdez, Melin et al. 2010), it was reported that empirical analysis using both types of membership functions concluded that trapezoidal membership functions give better performance over Gaussian ones. Hence, this work adopts the Mamdani type fuzzy inference system with the centre of gravity and trapezoidal membership functions to obtain a suitable performance.

Sixth, tuning of the fuzzy inference system can also be an issue. For example, triangular or Gaussian membership functions can be used instead of trapezoid membership functions. Similarly, the centre of gravity (COG) defuzzification method can be replaced with the maximum method. The performance of the fuzzy inference system can also be enhanced by changing the rule premises or actions, changing the centers of membership functions for input and/or output linguistic variables, or adding additional linguistic terms to the fuzzy variables. However, the most widely adopted design choices are selected and used successfully so as to propose a functional and efficient fuzzy inference system.

Lastly, the choices of efficiency and performance metrics can also pose as threats. Other metrics that evaluate the efficiency and performance utilizing the internal algorithm structure can exist. However, this work adopts the generated size of mixed strength *t*-way test suites for efficiency and generation time for performance because these metrics are well-known in the literature (i.e., for mixed strength *t*-way test suite generation).

4.7 Chapter Summary

To sum up, this chapter has provided the overall evaluation process of the ATLBO for the optimal generation of mixed strength *t*-way test suites. In the beginning, the chapter has elaborated how the experimental setup is laid out to achieve the predefined goals for the effective evaluation of ATLBO. In the three-part experiments, the first part adopts a set of CAs and VCAs to highlight the time and size performance of ATLBO and the original TLBO. The second part benchmarks the generated test suite sizes of ATLBO against the results obtained from the implementation of the original TLBO, as well as against the results of other meta-heuristic-based strategies adopted from a high-impact journal paper. The third part presents the statistical analysis of all the obtained results with 95% confidence level and 90% confidence level utilizing the Wilcoxon Rank Sum test. The chapter furnishes details about the control parameters for all the competing meta-heuristic-based strategies and the experimental platform.

The chapter then discusses the performance of ATLBO based on the experimental observations. Regarding the first part of the experiments, ATLBO has mostly outperformed TLBO in terms of the best and mean test suite sizes. However, TLBO outperformed ATLBO in terms of test suite generation time. In the second part of the experiments, ATLBO showed competitive performance against the two meta-heuristic based strategies (DPSO and SA), whereas outperformed all other referenced strategies in terms of test suite sizes. The tendency of ATLBO towards exploration and exploitation is also tracked. It is observed that ATLBO favours exploration with the increase in *p*, *t* and sub-configurations for VCAs. The Wilcoxon Rank Sum tests of the obtained results showed significant statistical performance of ATLBO against its counterparts. The null hypothesis was favoured in most tests for ATLBO against all other strategies except only DPSO. Finally, the chapter identifies the threats to validities encountered by the experiments. The threats include the choice of benchmarks, comparison with the selected strategies, twice fitness function evaluations per iteration in TLBO compared to once in ATLBO, randomness of search operators, choice of fuzzy implementation, tuning the inference system and the choices of efficiency and performance metrics. Detailed steps are undertaken to reduce the effects of all the threats (total seven) on the results. After completing the evaluation process of the ATLBO based on the experimental results, the next chapter concludes this work and provides possible future directions.

CHAPTER 5

CONCLUSION AND FUTURE WORK

The previous chapter has gauged performance and effectiveness of ATLBO on a number of experiments against the original TLBO and other meta-heuristic-based *t*-way test suite generation strategies. Based on all the presented materials in the earlier chapters, this chapter summarizes the impact of the proposed ATLBO algorithm for the problem of generating mixed strength *t*-way test suite with directions for future work.

5.1 Objectives Revisited

The aim of this research work was to design, implement and evaluate ATLBO for addressing the problem of mixed strength *t*-way test suite generation. The objectives of this research effort for fulfilling the stated aim were as follow:

- To design a new variant of TLBO called ATLBO based on a Mamdani-type fuzzy inference system for adaptively selecting exploitation (i.e., local search) and exploration (i.e., global search).
- To implement ATLBO for addressing the problem of generating both uniform and mixed strength *t*-way test suites.
- To evaluate the performance of ATLBO in terms of generated test suite sizes against the original TLBO and other meta-heuristic algorithms.

Addressing the first objective, a new improved variant of TLBO called adaptive TLBO (ATLBO) is designed. A Mamdani-type fuzzy inference system is introduced in ATLBO so as to further improve efficiency and stability in search of original TLBO. Using this fuzzy system, ATLBO summons, based on the current search requirement,

either teacher phase (i.e., global search) or learner phase (i.e., local search) per iteration rather than both as in original TLBO.

The proposed Mamdani-type fuzzy inference system of ATLBO has three input parameters: *Quality Measure* (Q_m), *Intensification Measure* (I_m), and *Diversification Measure* (D_m) and one output parameter *Selection*. Trapezoidal membership functions are used to fuzzify these linguistic variables. The rule based of the system is composed of four fuzzy linguistic rules with max-min fuzzy inference method. Finally, the center of gravity (COG) is used as the defuzzification method to obtain the *Selection* as a single value crisp output. This value acts as an intermittent switch in ATLBO that decides when to perform global search or when to perform local search.

The successful implementation of ATLBO for mixed strength test suite generation satisfies the key aspect of the second objective. ATLBO offers the first more efficient and completely adaptive strategy for the problem of generating mixed strength test suite with interaction strength support of $t = 4$. The ATLBO strategy accepts test configurations for software-under-test in the form of covering array notation as input. The strategy automatically minimizes the test suite after processing the test specification requirement.

The ATLBO strategy incorporates an efficient interaction elements generation algorithm in order to generate and search for the required interaction elements. The algorithm adopts hash map approach instead of the array list. With this approach, the algorithm is able to generate interaction elements based on the given number of input parameters, their values and interaction strength faster. Moreover, searching for the interaction elements is now also faster owing to the implementation of the algorithm in the proposed strategy.

Concerning the final objective, ATLBO has been successfully subjected to a wide range of well-known benchmarking experiments so as to highlight its performance for optimal test suite generation. The evaluation against existing state-of-the-art meta-heuristic based strategies has seamlessly revealed the size performance of ATLBO. In the experimentation, the results of ATLBO are successfully compared against the results of other meta-heuristic algorithms as far as test suite sizes are concerned. Experimental

results of the ATLBO have been encouraging as it has obtained several new minimal test suite sizes.

Considering interaction test configurations represented using CAs and VCAs with higher interaction strength $t \geq 2$, the ATLBO often generates minimal test suite sizes. For uniform test suites, ATLBO obtains 70.37% (i.e., 38 out of total 54 entries) best sizes and 61.11% (i.e., 33 out of total 54 entries) mean best sizes for all the benchmarked experiments reported in Table 4.2 and Tables 4.3-4.8. In the case of mixed strength test configurations represented using VCAs, the ATLBO strategy consistently produces best optimal test suite sizes. For mixed strength test suites, ATLBO obtains 53.57% (i.e., 15 out of total 28 entries) best sizes and 64.29% (i.e., 18 out of total 28 entries) mean best sizes (see Table 4.2 and Tables 4.6-4.8). Overall, ATLBO generates 53.42% (i.e., 39 out of total 73 entries) new mean best sizes as provided in Table 4.2 and Tables 4.3-4.8. For each experiment, the distribution pattern of ATLBO's exploration and exploitation operations is reported to determine how much it favors each operation. Statistical analyses reported in Tables 4.9-4.15 confirm the strong statistical performance of ATLBO against most existing strategies for *t*-way test suite generation. Based on these results, this research work concludes that the proposed ATLBO algorithm is another useful alternative for addressing the problem of generating both uniform and mixed strength *t*-way test suites.

In essence, the main focus of this thesis is to design ATLBO and combine it with the interaction elements generation algorithm so as to optimize mixed strength interaction test suites. ATLBO with its Mamdani-type fuzzy inference system successfully improves the efficiency and stability in search of its predecessor i.e., TLBO. The implementation of ATLBO is successful for generating optimal uniform and mixed strength *t*-way test suites for highly configurable software systems.

5.2 Contributions

In a nutshell, the earlier discussion relates the main contribution of ATLBO to its ability for generating minimal uniform and mixed strength *t*-way test suites. The research contributions of this work can be highlighted from different aspects as follows:

- ATLBO as an improved variant of original TLBO employs its Mamdani-type fuzzy inference system to adaptively select either exploration or exploitation, and hence, explores the search space more efficiently for optimal test suites.

- ATLBO introduces the first *parameter-free* meta-heuristic strategy in the literature for addressing the problem of generating mixed strength test suites with higher interaction strength $t = 4$.

- ATLBO contributes to the benchmarking test configurations available in the published literature with 39 new mean best test suite sizes.

5.3 Future Work

Given the competitive performance of ATLBO for mixed strength test suites, its adaptation for the generation of other types of mathematical objects is considered as the scope for future work. Specifically, the applicability of ATLBO will be investigated for the generation of cost-aware coverings arrays (CTCAs), constrained covering arrays (CCAs) and sequence covering arrays (SCAs).

Implementing ATLBO for CTCAs is one area for exploration in the future. CTCAs are recently introduced t-way testing objects that represent an interesting and novel research direction. The ATLBO strategy can be modified to compute CTCA for a given cost function associated with the CA. The fitness function, in this case, will be comprised of two objectives, namely covering all required t-way interaction elements and minimizing the cost function. Further minimization can be achieved as CTCAs ensure the accumulation of test cases with minimum possible costly interaction elements.

The generation of CCAs particularly for software product line will be undertaken as a future endeavor. Constrained interaction testing is gaining much research attention because most of the modern day software systems are subjected to constraints. A naive way of handling constraints is to exclude them from the final test suites. ATLBO will be combined with Choco Solver (Prud'homme, Fages et al. 2017) for handling constraints.

Incorporating support for generating SCAs via ATLBO could be another area for exploration. Sequences of input parameters in some domain implementations such as

Graphical User Interfaces (GUIs) do matter. Therefore, it is desirable that the proposed algorithm be able to generate sequence-based *t*-way test suites.

Currently, ATLBO only supports automated test suite generation. The addition of automated test execution can further improve the applicability of ATLBO. Test cases generated by ATLBO can be automatically translated into the actual executable form using some scripting language. The burden of test engineers of dealing with complex manual test execution can be alleviated by incorporating this useful feature in ATLBO.

Finally, the applicability of ATLBO with its powerful search ability will be investigated for other related optimization problems. Some of these problems include wireless sensor network localization and generation of Substitution boxes (S-boxes) in contemporary symmetric ciphers. Similarly, ATLBO will be used for solving search-based optimization problems in software engineering such as software module clustering problem, software effort estimation models and software redundancy reduction.

REFERENCES

Afzal, W., R. Torkar and R. Feldt (2009). "A Systematic Review of Search-based Testing for Non-functional System Properties." Information and Software Technology 51(6): 957-976.

Ahmed, B. S. (2016). "Test Case Minimization Approach using Fault Detection and Combinatorial Optimization Techniques for Configuration-aware Structural Testing." Engineering Science and Technology, an International Journal 19(2): 737-753.

Ahmed, B. S., T. S. Abdulsamad and M. Potrus (2015). "Achievement of Minimized Combinatorial Test Suite for Configuration-aware Software Functional Testing using the Cuckoo Search Algorithm." Information and Software Technology 66: 13-29.

Ahmed, B. S., L. M. Gambardella, W. Afzal and K. Z. Zamli (2017). "Handling Constraints in Combinatorial Interaction Testing in the Presence of Multi Objective Particle Swarm and Multithreading." Information and Software Technology 86: 20-36.

Ahmed, B. S. and K. Z. Zamli (2010). PSTG: A *t*-way Strategy Adopting Particle Swarm Optimization. Proceedings of the 4th Asia International Conference on Mathematical /Analytical Modelling and Computer Simulation, IEEE.

Ahmed, B. S. and K. Z. Zamli (2011a). Comparison of Metahuristic Test Generation Strategies based on Interaction Elements Coverage Criterion. Proceedings of the IEEE Symposium on Industrial Electronics and Applications, IEEE.

Ahmed, B. S. and K. Z. Zamli (2011b). "A Review of Covering Arrays and their Application to Software Testing." Journal of Computer Science 7(9): 1375-1385.

Ahmed, B. S. and K. Z. Zamli (2011c). "A Variable-Strength Interaction Test Suites Generation Strategy using Particle Swarm Optimization." Journal of Systems and Software 84(12): 2171-2185.

Ahmed, B. S., K. Z. Zamli and C. P. Lim (2012). "Application of Particle Swarm Optimization to Uniform and Variable Strength Covering Array Construction." Applied Soft Computing 12(4): 1330-1347.

Ahmed, B. S., K. Z. Zamli and C. P. Lim (2012). "Constructing a *t*-way Interaction Test Suite using the Particle Swarm Optimization Approach." International Journal of Innovative Computing, Information and Control 8(1): 431-452.

Alsariera, Y. A. and K. Z. Zamli (2015). "A Bat-inspired Strategy for *t*-way Interaction Testing." Advanced Science Letters 21(7): 2281-2284.

Alsewari, A. R. A. and K. Z. Zamli (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.

Ameli, K., A. Alfi and M. Aghaebrahimi (2016). "A Fuzzy Discrete Harmony Search Agorithm Applied to Annual Cost Reduction in Radial Distribution Systems." Engineering Optimization 48(9): 1529-1549.

Arifovic, J. (1996). "The Behavior of the Exchange Rate in the Genetic Algorithm and Experimental Economies." Journal of Political Economy 104(3): 510-541.

Avila-George, H., J. Torres-Jimenez, L. Gonzalez-Hernandez and V. Hernandez (2013). "Metaheuristic Approach for Constructing Functional Test-suites." IET Software 7(2): 104-117.

Avila-George, H., J. Torres-Jimenez and I. Izquierdo-Marquez (2018). "Search-Based Software Engineering for Constructing Covering Arrays." IET Software 12(4): 324-332.

Avila, C. and F. Valdez (2015). An Improved Simulated Annealing Algorithm for the Optimization of Mathematical Functions. Design of Intelligent Systems Based on Fuzzy Logic, Neural Networks and Nature-Inspired Optimization. P. Melin, O. Castillo and J. Kacprzyk. Cham, Springer**:** 241-251.

Beasley, D., R. Martin and D. Bull (1993). "An Overview of Genetic Algorithms: Part 1. Fundamentals." University Computing 15(2): 56-69.

Biswas, S., S. Kundu, D. Bose and S. Das (2012). Cooperative Co-Evolutionary Teaching-Learning based Algoritm with a Modified Exploration Strategy for Large Scale Global Optimization. International Conference on Swarm, Evolutionary, and Memetic Computing, Springer.

Boussaïd, I., J. Lepagnot and P. Siarry (2013). "A Survey on Optimization Meta-heuristics." Information Sciences 237: 82-117.

Bryce, R. and C. Colbourn (2007). One-Test-at-a-Time Heuristic Search for Interaction Test Suites. Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, ACM.

Burnstein, I. (2006). Practical Software Testing: A Process-oriented Approach, Springer Science & Business Media.

Busetti, F. (2003). "Simulated Annealing Overview." World Wide Web URL www. geocities. com/francorbusetti/saweb. pdf.

Camastra, F., A. Ciaramella, V. Giovannelli, M. Lener, V. Rastelli, A. Staiano, G. Staiano and A. Starace (2015). "A Fuzzy Decision System for Genetically Modified Plant Environmental Risk Assessment using Mamdani Inference." Expert Systems with Applications 42(3): 1710-1716.

Castillo, O., A. Meléndez, P. Melin, L. Astudillo and C. Sánchez (2015). Optimization of Reactive Fuzzy Controllers for Mobile Robots based on the Chemical Reactions Algorithm. Design of Intelligent Systems based on Fuzzy Logic, Neural Networks and Nature-inspired Optimization. P. Melin, O. Castillo and J. Kacprzyk, Springer**:** 253-266.

Castillo, O., H. Neyoy, J. Soria, P. Melin and F. Valdez (2015). "A New Approach for Dynamic Fuzzy Logic Parameter Tuning in Ant Colony Optimization and its Aplication in Fuzzy Control of a Mobile Robot." Applied Soft Computing 28: 150-159.

Chen, X., Q. Gu, A. Li and D. Chen (2009). Variable Strength Interaction Testing with an Ant Colony System Approach. Proceedings of the 16th Asia-Pacific Software Engineering Conference, IEEE

Cheng, M.-Y. and D. Prayogo (2014). "Symbiotic Organisms Search: A New Metaheuristic Optimization Algorithm." Computers & Structures 139: 98-112.

Cheng, M.-Y. and D. Prayogo (2016). "Fuzzy Adaptive Teaching–learning-based Optimization for Global Numerical Optimization." Neural Computing and Applications 29(2): 309-327.

Cheng, M. Y., P. M. Firdausi and D. Prayogo (2014). "High-performance Concrete Compressive Strength Prediction using Genetic Weighted Pyramid Operation Tree (GWPOT)." Engineering Applications of Artificial Intelligence 29: 104-113.

Cheng, M. Y., D. Prayogo and Y. W. Wu (2014). "Novel Genetic Algorithm-based Evolutionary Support Vector Machine for Optimizing High-performance Concrete Mixture." Journal of Computing in Civil Engineering 28(4).

Cheng, M. Y., D. K. Wibowo, D. Prayogo and A. F. V. Roy (2015). "Predicting Productivity Loss Caused by Change Orders using the Evolutionary Fuzzy Support Vector Machine Inference Model." Journal of Civil Engineering and Management 21(7): 881-892.

Chikh, M. A. A., I. Belaidi, S. Khelladi, J. Paris, M. Deligant and F. Bakir (2018). "Efficiency of Bio-and Socio-inspired Optimization Algorithms for Axial Turbomachinery Design." Applied Soft Computing 64: 282-306.

Cohen, D. M., S. R. Dalal, M. L. Fredman and G. C. Patton (1997). "The AETG System: An Approach to Testing based on Combinatorial Design." IEEE Transactions on Software Engineering 23(7): 437–443.

Cohen, M. B. (2004). Designing Test Suites For Software Interaction Testing Doctor of Philosophy (PhD) Thesis, University of Auckland.

Cohen, M. B., C. J. Colbourn and A.C.H.Ling (2003). Augmenting Simulated Annealing to Build Interaction Test Suite. Proceedings of the14th International Symposium on Software Reliability Engineering, IEEE.

Cohen, M. B., C. J. Colbourn and A. C. H. Ling (2008). "Constructing Strength Three Covering Arrays with Augmented Annealing." Discrete Mathematics 308(13): 2709-2722.

Cohen, M. B., M. B. Dwyer and J. Shi (2007). Interaction Testing of Highly-configurable Systems in the Presence of Constraints. Proceedings of the International Symposium on Software Testing and Analysis, ACM.

Cohen, M. B., P. B. Gibbons, W. B. Mudgridge, C. J. Colbourn and J. S. Collofello (2003a). Variable Strength Interaction Testing of Components. Proceedings of the 27th Annual International Conference on Computer Software and Applications, IEEE.

Cohen, M. B., P. B. Gibbons, W. B. Mugridge and C. J. Colbourn (2003b). Constructing Test Suites for Interaction Testing. Proceedings of the 25th International Conference on Software Engineering, IEEE: 38-48.

Colbourn, C. J. and J. H. Dinitz (2006). Handbook of Combinatorial Designs, CRC Press.

Cordón, O. (2011). "A Historical Review of Evolutionary Learning Methods for Mamdani-type Fuzzy Rule-based Systems: Designing Interpretable Genetic Fuzzy Systems." International Journal of Approximate Reasoning 52(6): 894-913.

Dadios, E. P. (2012). Fuzzy Logic–controls, Concepts, Theories and Applications, InTechOpen.

Demiroz, G. and C. Yilmaz (2016). "Using Simulated Annealing for Computing Cost-aware Covering Arrays." Applied Soft Computing 49: 1129-1144.

Draa, A. (2015). "On the Performances of the Flower Pollination Algorithm- Qualitative and Quantitative Analyses." Applied Soft Computing 34: 349-371.

Eaarts, E. and J. Korst (1989). Simulated Annealing and Boltzman Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing, John Wiley and Sons.

Esfandyari, S. and V. Rafe (2018). "A Tuned Version of Genetic Algorithm for Efficient Test Suite Generation in Interactive *t*-way Testing Strategy." Information and Software Technology 94: 165-185.

Federer, W. T. and J. P. Mandeli (1986). "Orthogonal F-rectangles, Orthogonal Arrays, and Codes." Journal of Combinatorial Theory, Series A 43(2): 149-164.

Fister, I., I. Fister, X. S. Yang, S. Fong and Y. Zhuang (2014). Bat Algorithm: Recent Advances. Proceedings of the 15th International Symposium on Computational Intelligence and Informatics, IEEE.

Gandomi, A. H. and A. R. Kashani (2018). "Construction Cost Minimization of Shallow Foundation using Recent Swarm Intelligence Techniques." IEEE Transactions on Industrial Informatics 14(3): 1099-1106.

Garvin, B., M. Cohen and M. Dwyer (2011). "Evaluating Improvements to a Meta-heuristic Search for Constrained Interaction Testing." Empirical Software Engineering 16: 61-102.

Garvin, B. J. and M. B. Cohen (2011). Feature Interaction Faults Revisited: An Exploratory Study. The IEEE 22nd International Symposium on Software Reliability Engineering, IEEE: 90-99.

Garvin, B. J., M. B. Cohen and M. B. Dwyer (2009). An Improved Meta-heuristic Search for Constrained Interaction Testing. Proceedings of the 1st International Symposium on Search Based Software Engineering, IEEE.

Geem, Z. W. and J. H. Kim (2001). "A New Heuristic Optimization Algorithm: Harmony Search." Simulation 76(2): 60-68.

Glover, F. (1989). "Tabu Search-Part I." ORSA Journal on Computing 1(3): 190-206.

Gonzalez-Hernadez, L. (2015). "New Bounds for Mixed Covering Arrays in *t*-way Testing with Uniform Strength " Information and Software Technology 59: 17-32.

Gonzalez-Hernandez, L., N. Rangel-Valdez and J. Torres-Jimenez (2010). Construction of Mixed Covering Arrays of Variable Strength using a Tabu Search Approach. Proceedings of the International Conference on Combinatorial Optimization and Applications, Springer.

Gonzalez-Hernandez, L., N. Rangel-Valdez and J. Torres-Jimenez (2012). "Construction of Mixed Covering Arrays of Strengths 2 through 6 using a Tabu Search Approach." Discrete Mathematics, Algorithms and Applications 4(3).

Gonzalez-Hernandez, L. and J. Torres-Jimenez (2010). MiTS: A New Approach of Tabu Search for Constructing Mixed Covering Arrays. The Springer 9th Mexican International Conference on Artificial Intelligence in Soft Computing: Part II, Springer**:** 382-393.

Grefenstette, J. J. (1986). "Optimization of Control Parameters for Genetic Algorithms." IEEE Transactions on Systems, Man and Cybernetics 16(1): 122-128.

Hartman, A. and L. Raskin (2004). "Problems and Algorithms for Covering Arrays." Discrete Mathematics 284(1-3): 149-156.

Hass, A. M. (2014). Guide to Advanced Software Testing, Artech House.

Holland, J. H. (1975). Adaptation in Natural and Artificial Systems., University of Michigan Press.

Hoseini, M., H. Hosseinpour and B. Bastaee (2014). "A New Multi Objective Optimization Approach in Distribution Systems." Optimization Letters 8(1): 181-199.

Huang, G.-Q., W.-J. Zhao and Q.-Q. Lu (2013). "Bat Algorithm with Global Convergence for Solving Large-scale Optimization Problem." Jisuanji Yingyong Yanjiu 30(5): 1323-1328.

Huang, J., L. Gao and X. Li (2015). "An Effective Teaching-Learning based Cuckoo Search Algorithm for Parameter Optimization Problems in Structure Designing and Machining Processes." Applied Soft Computing 36: 349-356.

Iancu, I. (2012). A Mamdani-type Fuzzy Logic Controller. Fuzzy Logic-Controls, Concepts, Theories and Applications, InTech.

Jia, Y., M. B. Cohen, M. Harman and J. Petke (2015). Learning Combinatorial Interaction Test Generation Strategies Using Hyperheuristic Search. Proceedings of the 37th International Conference on Software Engineering, IEEE.

Jiang, X. and J. Zhou (2013). Hybrid DE-TLBO for Solving Short Term Hydro-thermal Optimal Scheduling with Incommensurable Objectives. Proceedings of the 32nd Chinese Control Conference, IEEE.

Jorgensen, P. C. (2016). Software Testing: A Craftsman's Approach, CRC press.

Kacker, R. N., D. R. Kuhn, Y. Lei and J. F. Lawrence (2013). "Combinatorial Testing for Software: An Adaptation of Design of Experiments." Measurement: Journal of the International Measurement Confederation 46(9): 3745-3752.

Kitsos, P., D. E. Simos, J. Torres-Jimenez and A. G. Voyiatzis (2015). Exciting FPGA Cryptographic Trojans using Combinatorial Testing. The IEEE 26th International Symposium on Software Reliability Engineering, IEEE: 69-76.

Kuhn, D. R., J. M. Higdon, J. F. Lawrence, R. N. Kacker and Y. Lei (2012). Combinatorial Methods for Events Sequence Testing. Proccedings 5th International Conference on Software Testing, Verification and Validation, IEEE.

Kuhn, D. R. and V. Okum (2006). Pseudo-exhaustive Testing for Software. Proceedings of the 30th Annual IEEE/NASA Software Engineering Workshop, IEEE.

Kuhn, D. R., D. R. Wallace and A. M. Gallo (2004). "Software Fault Interactions and Implications for Software Testing." IEEE Transactions on Software Engineering 30(6): 418-421.

Kuliamin, V. V. and A. A. Petukhov (2011). "A Survey of Methods for Constructing Covering Arrays." Programming and Computer Software 37(3): 121-146.

Lee, K. Y. and J.-B. Park (2006). Application of Particle Swarm Optimization to Economic Dispatch Problem: Advantages and Disadvantages. Proceedings of the Power Systems Conference and Exposition, IEEE.

Lei, D., L. Gao and Y. Zheng (2018). "A Novel Teaching-learning-based Optimization Algorithm for Energy-efficient Scheduling in Hybrid Flow Shop." IEEE Transactions on Engineering Management 65(2): 330-340.

Lei, Y., R. Kacker, D. R. Kuhn, V. Okun and J. Lawrence (2008). "IPOG-IPOG-D: Efficient Test Generation for Multi-way Combinatorial Testing." Software Testing, Verification & Reliability 18(3): 125-148.

Lei, Y. and K.-C. Tai (1998). In-Parameter-Order: A Test Generation Strategy for Pairwise Testing. The 3rd IEEE International Symposium on High-Assurance Systems Engineering, IEEE: 254-261.

Leyden, J. (2012). Symantec Update Killed Business PCs in *Three*-way Software Prang. The Register.

Lim, W. H. and N. A. M. Isa (2014). "Teaching and Peer-Learning Particle Swarm Optimization." Applied Soft Computing 18: 39-58.

Lin, J., C. Luo, S. Cai, K. Su, D. Hao and L. Zhang (2015). TCA: An Efficient Two-mode Meta-heuristic Algorithm for Combinatorial Test Generation. Proceedings of the 30th IEEE/ACM International Conference on Automated Software Engineering, IEEE.

M. Črepinšek, S.-H. Liu and M. Mernik (2013). "Exploration and Exploitation in Evolutionary Algorithms: A Survey." ACM Computing Surveys 45(3).

M. Črepinšek, S.-H. Liu, L. Mernik and M. Mernik (2015). "Is a Comparison of Results Meaningful from the Inexact Replications of Computational Experiments?" Soft Computing 20(1): 223-235.

M. Črepinšek, S.-H. Liu and M. Mernik (2014). "Replication and Comparison of Computational Experiments in Applied Evolutionary Computing: Common Pitfals and Guidelines to avoid them." Applied Soft Computing 19: 161-170.

M. Mernik, S.-H. Liu, D. Karaboga and M. Črepinšek (2015). "On Clarifying Misconceptions when Comparing Variants of the Artificial Bee Colony Algorithm by Offering a New Implementation." Information Sciences 291: 115-127.

Mahmoud, T. and B. S. Ahmed (2015). "An Efficient Strategy for Covering Array Construction with Fuzzy Logic-based Adaptive Swarm Optimization for Software Testing Use." Expert Systems with Applications 42(22): 8753-8765.

Mamdani, E. and S. Assilian (1975). "An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller." International Journal of Man-Machine Studies 7(1): 1-13.

Mandal, B. and P. K. Roy (2013). "Optimal Reactive Power Dispatch using Quasi-Oppositional Teaching Learning based Optimization." International Journal of Electrical Power & Energy Systems 53: 123-134.

Mirjalili, S., S. Z. Mohd Hashim and H. Moradian Sardroudi (2012). "Training Feedforward Neural Networks using Hybrid Particle Swarm Optimization and Gravitational Search Algorithm." Applied Mathematics and Computation 218(22): 11125-11137.

Mohd Hazli, M. Z. and K. Z. Zamli (2013). "Implementing a *t*-Way Test Generation Strategy using Bees Algorithm." International Journal of Advances in Soft Computing & Its Applications 5(3): 116-126.

Mohd Hazli, M. Z., K. Z. Zamli and R. R. Othman (2012). Sequence-based Interaction Testing Implementation using Bees Algorithm. Proceedings of the IEEE Symposium on Computers and Informatics, IEEE.

Myers, G. J., C. Sandler and T. Badgett (2011). The Art of Software Testing, John Wiley & Sons.

Neyoy, H., O. Castillo and J. Soria (2013). Dynamic Fuzzy Logic Parameter Tuning for ACO and its Application in TSP Problems. Recent Advances on Hybrid Intelligent Systems, Springer**:** 259-271.

Nie, C. and H. Leung (2011). "A Survey of Combinatorial Testing." ACM Computing Surveys 43(2): 1-29.

Niknam, T., R. Azizipanah-Abarghooee and J. Aghaei (2013). "A New Modified Teaching-learning Algorithm for Reserve Constrained Dynamic Economic Dispatch." IEEE Transactions on Power Systems 28(2): 749-763.

Niu, P., Y. Ma and S. Yan (2018). "A Modified Teaching–learning-based Optimization Algorithm for Numerical Function Optimization." International Journal of Machine Learning and Cybernetics: 1-15.

Niu, X., c. N, H. K. N. Leung, Y. Lei, X. Wang, J. Xu and Y. Wang (2018). "An Interleaving Approach to Combinatorial Testing and Failure-inducing Interaction Identification." IEEE Transactions on Software Engineering 13(9): 1-33.

Nurmela, K. J. (2004). "Upper Bounds for Covering Arrays by Tabu Search." Discrete Applied Mathematics 138(1-2): 143-152.

Pappis, C. P. and C. I. Siettos (2014). Fuzzy Reasoning. Search Methodologies. E. K. Burke and G. Kendall, Springer**:** 519-556.

Pedersen, M. E. H. (2010). "Good Parameters for Particle Swarm Optimization." Hvass Lab., Copenhagen, Denmark, Tech. Rep. HL1001.

Pérez, J., F. Valdez and O. Castillo (2015). A New Bat Algorithm with Fuzzy Logic for Dynamical Parameter Adaptation and its Applicability to Fuzzy Control Design. Fuzzy Logic Augmentation of Nature-Inspired Optimization Metaheuristics, Springer**:** 65-79.

Pérez, J., F. Valdez and O. Castillo (2017). Modification of the Bat Algorithm using Type-2 Fuzzy Logic for Dynamical Parameter Adaptation. Nature-Inspired Design of Hybrid Intelligent Systems, Springer**:** 343-355.

Pham, D. T., A. Ghanbarzadeh, E. Koç, S. Otri, S. Rahim and M. Zaidi (2006). The Bees Algorithm — A Novel Tool for Complex Optimisation Problems. Intelligent Production Machines and Systems. D. T. Pham, E. E. Eldukhri and A. J. Soroka, Elsevier Science**:** 454-459.

Prud'homme, C., J.-G. Fages and X. Lorca (2017). Choco Documentation.

Rao, R. (2016). "Jaya: A Simple and New Optimization Algorithm for Solving Constrained and Unconstrained Optimization Problems." International Journal of Industrial Engineering Computations 7(1): 19-34.

Rao, R. and V. Patel (2012). "An Elitist Teaching-learning-based Optimization Algorithm for Solving Complex Constrained Optimization Problems." International Journal of Industrial Engineering Computations 3(4): 535-560.

Rao, R. V., V. J. Savsani and D. P. Vakharia (2011). "Teaching-learning-based Optimization: A Novel Method for Constrained Mechanical Design Optimization Problems." CAD Computer Aided Design 43(3): 303-315.

Rao, R. V., V. J. Savsani and D. P. Vakharia (2012). "Teaching-learning-based Optimization: An Optimization Method for Continuous Non-linear Large Scale Problems." Information Sciences 183(1): 1-15.

Robinson, J. and Y. Rahmat-Samii (2004). "Particle Swarm Optimization in Electromagnetics." IEEE Transactions on Antennas and Propagation 52(2): 397-407.

Rodriguez-Cristerna, A. and J. Torres-Jimenez (2012). "A Simulated Annealing with Variable Neighborhood Search Approach to Construct Mixed Covering Arrays." Electronic Notes in Discrete Mathematics 39: 249-256.

Rodriguez-Cristerna, A., J. Torres-Jimenez, W. Gómez and W. C. A. Pereira (2015). "Construction of Mixed Covering Arrays Using a Combination of Simulated Annealing and Variable Neighborhood Search." Electronic Notes in Discrete Mathematics 47: 109-116.

Rodriguez-Tello, E. and J. Torres-Jimenez (2009). Memetic Algorithms for Constructing Binary Covering Arrays of Strength Three. International Conference on Artificial Evolution (Evolution Artificielle), Springer.

Roeva, O., S. Fidanova and M. Paprzycki (2013). Influence of the Population Size on the Genetic Algorithm Performance in Case of Cultivation Process Modelling. Proceedings of the Federated Conference on Computer Science and Information Systems IEEE.

Sabharwal, S., P. Bansal and N. Mittal (2016). "Construction of *t*-way Covering Arrays using Genetic Algorithm." International Journal of System Assurance Engineering and Management 8(2): 264-274.

Satapathy, S. C. and A. Naik (2013). "Cooperative Teaching-Learning based Optimization for Global Function Optimization." International Journal of Applied Research on Information Technology and Computing 4(1): 1-17.

Shabanpour-Haghighi, A., A. R. Seifi and T. Niknam (2014). "A Modified Teaching–Learning Based Optimization for Multi-Objective Optimal Power Flow Problem." Energy Conversion and Management 77: 597-607.

Sheikhan, M. and S. A. Ghoreishi (2013). "Application of Covariance Matrix Adaptation-evolution Strategy to Optimal Control of Hepatitis B Infection." Neural Computing and Applications 23(3-4): 881-894.

Shiba, T., T. Tsuchiya and T. Kikuno (2004). Using Artificial Life Techniques to Generate Test Cases for Combinatorial Testing. Proceedings of the 28th Annual International Conference on Computer Software and Applications, IEEE

Singh, R., H. Chaudhary and A. K. Singh (2017). "Defect-free Optimal Synthesis of Crank-rocker Linkage using Nature-inspired Optimization Algorithms." Mechanism and Machine Theory 116: 105-122.

Solano-Aragón, C. and O. Castillo (2015). Optimization of Benchmark Mathematical Functions using the Firefly Algorithm with Dynamic Parameters. Fuzzy Logic Augmentation of Nature-Inspired Optimization Metaheuristics, Springer**:** 81-89.

Srivastava, P. R., P. Patel and S. Chatrola (2009). "Cause Effect Graph to Decision Table Generation." ACM SIGSOFT Software Engineering Notes 34(2): 1-4.

Sugeno, M. (1985). "An Introductory Survey of Fuzzy Control." Information Sciences 36(1): 59-83.

Talatahari, S., M. Kheirollahi, C. Farahmandpour and A. H. Gandomi (2013). "A Multi-stage Particle Swarm for Optimum Design of Truss Structures." Neural Computing and Applications 23(5): 1297-1309.

Talbi, E.-G. (2009). Metaheuristics: From Design to Implementation, John Wiley & Sons.

Timaná-Peña, J. A., C. A. Cobos-Lozada and J. Torres-Jimenez (2016). "Metaheuristic Algorithms for Building Covering Arrays: A Review." Revista Facultad de Ingeniería 25(43): 31-45.

Torres-Jimenez, J. and E. Rodriguez-Tello (2010). Simulated Annealing for Constructing Binary Covering Arrays of Variable Strength. Proceedings of the IEEE Congress on Evolutionary Computation, IEEE.

Tuo, S., L. Yong and T. Zhou (2013). "An Improved Harmony Search based on Teaching-Learning Strategy for Unconstrained Optimization Problem." Mathematical Problems in Engineering.

Valdez, F., P. Melin and O. Castillo (2010). Fuzzy Control of Parameters to Dynamically Adapt the PSO and GA Algorithms. Proceedings of the IEEE International Conference on Fuzzy Systems, IEEE.

Valenzuela, L., F. Valdez and P. Melin (2017). Flower Pollination Algorithm with Fuzzy Approach for Solving Optimization Problems. Nature-Inspired Design of Hybrid Intelligent Systems, Springer**:** 357-369.

Walker Ii, R. A. and C. J. Colbourn (2009). "Tabu Search for Covering Arrays using Permutation Vectors." Journal of Statistical Planning and Inference 139(1): 69-80.

Wang, B.-C., H.-X. Li and Y. Feng (2018). "An Improved Teaching-Learning-Based Optimization for Constrained Evolutionary Optimization." Information Sciences 456: 131-144.

Williams, A. W. and R. L. Probert (1996). A Practical Strategy for Testing Pair-wise Coverage of Network Interfaces. Proceedings of the 7th International Symposium on Software Reliability Engineering, IEEE

Williams, A. W. and R. L. Probert (2001). A Measure for Component Interaction Test Coverage. Proceedings of the ACS /IEEE International Conference on Computer Systems and Applications, IEEE.

Wolpert, D. H. and W. G. Macready (1997). "No Free Lunch Theorems for Optimization." IEEE Transactions on Evolutionary Computation 1(1): 67-82.

Wu, H., C. Nie, F.-C. Kuo, H. Leung and C. J. Colbourn (2015). "A Discrete Particle Swarm Optimization for Covering Array Construction." IEEE Transactions on Evolutionary Computation 19(4): 575-591.

Xia, K., L. Gao, W. Li and K.-M. Chao (2014). "Disassembly Sequence Planning using a Simplified Teaching–learning-based Optimization Algorithm." Advanced Engineering Informatics 28(4): 518-527.

Yang, X.-S. (2010a). Nature-Inspired Metaheuristic Algorithm, Luniver Press.

Yang, X.-S. (2010b). A New Metaheuristic Bat-inspired Algorithm. Nature Inspired Cooperative Strategies for Optimization. J. R. González, D. A. Pelta, C. Cruz, G. Terrazas and N. Krasnogor, Springer: 65-74.

Yang, X.-S. and S. Deb (2009). Cuckoo Search via Lévy Flights. Proceedings of the World Congress on Nature & Biologically Inspired Computing, IEEE.

Yang, X. S., S. Deb and S. Fong (2013). "Metaheuristic Algorithms: Optimal Balance of Intensification and Diversification." Applied Mathematics & Information Sciences 8(3): 977-983.

Yen, J. and R. Langari (1999). Fuzzy Logic: Intelligence, Control, and Information, Prentice Hall Upper Saddle River, NJ.

Yilmaz, C., M. B. Cohen and A. Porter (2006). "Covering Arrays for Efficient Fault Characterization in Complex Configuration Spaces." IEEE Transactions on Software Engineering 32(1): 20-34.

Yilmaz, C., S. Fouch, M. B. Cohen, A. Porter, G. Demiroz and U. Koc (2014). "Moving Forward with Combinatorial Interaction Testing." Computer 47(2): 37-45.

Yin, Z., X. Ma, J. Zheng, Y. Zhou, L. N. Bairavasundaram and S. Pasupathy (2011). An Empirical Study on Configuration Errors in Commercial and Open Source Systems. Proceedings of the 23rd ACM Symposium on Operating Systems Principles, ACM: 159-172.

Zadeh, L. A. (1965). "Fuzzy Sets." Information and Control 8(3): 338-353.

Zamli, K. Z., B. Y. Alkazemi and G. Kendall (2016). "A Tabu Search Hyper-heuristic Strategy for t-way Test Suite Generation." Applied Soft Computing 44: 57-74.

Zimmermann, H.-J. (1996). Fuzzy Control. Fuzzy Set Theory—and Its Applications, Springer: 203-240.

Zou, F., L. Wang, X. Hei, D. Chen and B. Wang (2013). "Multi-Objective Optimization using Teaching-Learning based Optimization Algorithm." Engineering Applications of Artificial Intelligence 26(4): 1291-1300.

Zubrow, D. (2009). "IEEE Standard Classification for Software Anomalies." IEEE Computer Society.

APPENDIX A
LIST OF PUBLICATIONS

Published Journal Papers:

Zamli, K. Z., F. Din, S. Baharom and B. S. Ahmed (2017). "Fuzzy Adaptive Teaching Learning-based Optimization Strategy for the Problem of Generating Mixed Strength t-way Test Suites." Engineering Applications of Artificial Intelligence 59: 35-50.

Journal Impact Factor: 2.819 (Q1)

Zamli, K. Z., F. Din, G. Kendall and B. S. Ahmed (2017). "An Experimental Study of Hyper-heuristic Selection and Acceptance Mechanism for Combinatorial t-way Test Suite Generation." Information Sciences 399: 121-153.

Journal Impact Factor: 4.305 (Q1)

Zamli, K. Z., F. Din, B. S. Ahmed and M. Bures (2018). "A Hybrid Q-learning Sine-cosine-based Strategy for Addressing the Combinatorial Test Suite Minimization Problem." PLoS ONE 13(5).

Journal Impact Factor: 2.766 (Q1)

Din, F. and K. Z. Zamli (2018). "Hyper-Heuristic Based Strategy for Pairwise Test Case Generation." Advanced Science Letters 24(10): 7333-7338.

Published Conference Papers:

Din, F. and K. Z. Zamli (2017). Fuzzy Adaptive Teaching Learning-based Optimization Strategy for Pairwise Testing. Proceedings of the 7th International Conference on System Engineering and Technology, IEEE.

Din, F. and K. Z. Zamli (2018). Pairwise Test Suite Generation Using Adaptive Teaching Learning-Based Optimization Algorithm with Remedial Operator. Proceedings of the International Conference of Reliable Information and Communication Technology, Springer.

Din, F. and K. Z. Zamli (2018). Fuzzy Adaptive Teaching Learning-based Optimization Strategy for GUI Functional Test Cases Generation. The ACM 7th International Conference on Software and Computer Applications., ACM**:** 92-96.

Din, F., A. R. A. Alsewari and K. Z. Zamli (2017). A Parameter Free Choice Function based Hyper-heuristic Strategy for Pairwise Test Generation. Proceedings of the International Conference on Software Quality, Reliability and Security Companion, IEEE.

Zamli, K. Z., N. Safieny and F. Din (2018). Hybrid Test Redundancy Reduction Strategy based on Global Neighborhood Algorithm and Simulated Annealing. The ACM 7th International Conference on Software and Computer Applications., ACM**:** 87-91.

Accepted Conference Papers:

Zamli, K. Z., F. Din, N. Ramli and B. S. Ahmed (2018). Software Module Clustering based on the Fuzzy Adaptive Teaching Learning based Optimization Algorithm. Proceedings of the 2nd International Conference on Intelligent and Interactive Computing, Springer

APPENDIX B
BEST PAPER AWARD

APPENDIX C
MALAYSIAN INTERNATIONAL SCHOLARSHIP

KEMENTERIAN PENDIDIKAN TINGGI
MINISTRY OF HIGHER EDUCATION

Bahagian Biasiswa
Scholarship Division
Aras 2, No. 2, Menara 2, Jalan P5/6, Presint 5
62200 PUTRAJAYA
MALAYSIA

Tel : 603-8870 6000
Hotline : 603-8888 1616 / 6787
Faks : 603-8870 6839

Ruj. Kami : KPT. 600-4/1/12 Jld 3 BE 0271660 ( )
Tarikh : 6 September 2017

KEPADA SESIAPA YANG BERKENAAN

Tuan,

NAMA PELAJAR : FAKHRUD DIN
NEGARA : PAKISTAN
NO. PASPORT : BN7797082

Adalah dimaklumkan bahawa pelajar di atas telah ditawarkan biasiswa di bawah ***Malaysia International Scholarship (MIS)*** untuk pascasiswazah oleh Kerajaan Malaysia untuk melanjutkan pelajaran ke peringkat **PhD** dalam bidang **Sains Komputer** di **Universiti Malaysia Pahang (UMP)**. Tempoh penajaan pelajar berkuatkuasa mulai **01/09/2017 (Sem 1 2017/2018 – Sem 1 2018/2019 sahaja)** selama **3 semester tanpa pelanjutan**. Pelajar berkelayakan untuk menerima biasiswa (elaun sara hidup) daripada Kerajaan Malaysia berjumlah **RM18,000.00 setahun**. Selain daripada itu semua yuran pengajian kursus dan tiket penerbangan sekali pergi dan sekali balik bagi pelajar akan ditanggung oleh Kerajaan Malaysia. Walau bagaimanapun lain-lain bayaran seperti *caution fees/personal bond*, yuran asrama, insuran/yuran kesihatan, semua jenis cagaran, yuran peralatan, bayaran nota/ jurnal dan lain-lain yuran bukan akademik **perlu dijelaskan sendiri oleh pelajar**. **Tuntutan bayaran yuran pengajian hendaklah dibuat dalam tempoh satu bulan dari tarikh pelajar mendaftar diri** kepada:-

Ketua Setiausaha
Kementerian Pendidikan Tinggi
Bahagian Biasiswa
Aras 2, No.2, Menara 2
Jalan P5/6, Presint 5,
Pusat Pentadbiran Kerajaan Persekutuan
62200 PUTRAJAYA
(u.p: Pn. Hanizah Binti Hamdan)

Sekian, terima kasih.

'BERKHIDMAT UNTUK NEGARA'

Saya yang menurut perintah,

(HANIZAH BINTI HAMDAN)
Bahagian Biasiswa
b.p Ketua Setiausaha
Kementerian Pendidikan Tinggi
Emel : hanizah@mohe.gov.my
Tel : 03-8870 6387
Faks : 03-8870 6839

APPENDIX D
Q1 PAPERS

First Q1 Paper:

Engineering Applications of Artificial Intelligence 59 (2017) 35–50

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

Fuzzy adaptive teaching learning-based optimization strategy for the problem of generating mixed strength *t*-way test suites

Kamal Z. Zamli[a,*], Fakhrud Din[a], Salmi Baharom[b], Bestoun S. Ahmed[c]

[a] IBM Centre of Excellence, Faculty of Computer Systems and Software Engineering, Universiti Malaysia Pahang, Lebuhraya Tun Razak, 26300 Kuantan, Pahang Darul Makmur, Malaysia
[b] Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, Malaysia
[c] Department of Computer Science, Faculty of Electrical Engineering Czech Technical University Karlovo n´am. 13, 121 35 Praha 2, Czech Republic

ARTICLE INFO

Keywords:
Software testing
t-way testing
Teaching learning-based optimization algorithm
Mamdani fuzzy inference system

ABSTRACT

The teaching learning-based optimization (TLBO) algorithm has shown competitive performance in solving numerous real-world optimization problems. Nevertheless, this algorithm requires better control for exploitation and exploration to prevent premature convergence (i.e., trapped in local optima), as well as enhance solution diversity. Thus, this paper proposes a new TLBO variant based on Mamdani fuzzy inference system, called ATLBO, to permit adaptive selection of its global and local search operations. In order to assess its performances, we adopt ATLBO for the mixed strength *t*-way test generation problem. Experimental results reveal that ATLBO exhibits competitive performances against the original TLBO and other meta-heuristic counterparts.

1. Introduction

In the past decades, a few meta-heuristic algorithms have been proposed in scientific literature to address real-world optimization problems. These algorithms mainly comprises exploration and exploitation (or diversification and intensification) (Talbi, 2013). Exploration roams the random search space on a global scale (i.e., global search), whereas exploitation focuses on searching in a local region by exploiting the current suitable solution (i.e., local search). Overemphasizing exploration consumes significant computational resources and prevents convergence. Conversely, excessive exploitation tends to deny a diverse solution and may lead toward local optima. Most meta-heuristic algorithms introduce specific parameter controls to manage exploration and exploitation effectively. For example, genetic algorithm (GA) (Holland, 1975) exploits mutation and crossover rate; particle swarm optimization (PSO) (Kennedy and Eberhart, 1995) introduces inertia weight and social/cognitive parameters; harmony search (HS) (Geem, 2009) relies on the consideration rate of harmony memory and pitch adjustment; and ant colony optimization (ACO) (Dorigo et al., 1996) exploits evaporation rate, pheromone influence, and heuristic influence. Tuning the parameters accordingly ensures a suitable quality solution. However, the tuning of these parameters is often time consuming and problem specific because a single size is unavailable to fits all approaches.

The teaching learning-based optimization algorithm (TLBO) (Rao et al., 2011, 2012) adopts a simplistic approach of disregarding the control parameters (i.e., parameter free). TLBO specifically performs both global and local search sequentially per iteration to balance exploration and exploitation. Given that exploration and exploitation are dynamic in nature depending on the current search space region, any preset division between the two can be counter-productive and may lead to poor quality solutions. This paper addresses these issues through a new TLBO variant, adaptive TLBO (ATLBO) integrated with the Mamdani-type fuzzy inference system (Camastra et al., 2015; Cordón, 2011). ATLBO adaptively selects its local and global search operations. In order to assess its performances, we adopt ATLBO for the mixed strength *t*-way test generation problem.

Our contributions are summarized as follows:

- The novel ATLBO strategy based on the Mamdani-type fuzzy inference system is presented for exploration (i.e., global search) and exploitation (i.e., local search) selection.
- ATLBO is the first TLBO-variant strategy that addresses generation for both uniform and mixed-strength *t*-way test suite.

This study is organized as follows. Section 2 presents the theoretical framework that covers the generation problem of *t*-way test and its mathematical notation. Section 3 describes the related work. Section 4

* Corresponding author.
E-mail addresses: kamalz@ump.edu.my (K.Z. Zamli), fakhruddin@uom.edu.pk (F. Din), salmi@upm.edu.my (S. Baharom), bestoon82@gmail.com (B.S. Ahmed).

http://dx.doi.org/10.1016/j.engappai.2016.12.014
Received 8 September 2016; Received in revised form 2 December 2016; Accepted 15 December 2016
Available online 22 December 2016
0952-1976/

Second Q1 Paper:

Information Sciences 399 (2017) 121–153

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

An experimental study of hyper-heuristic selection and acceptance mechanism for combinatorial *t*-way test suite generation

Kamal Z. Zamli [a,*], Fakhrud Din [a], Graham Kendall [b], Bestoun S. Ahmed [c]

[a] IBM Centre of Excellence, Faculty of Computer Systems and Software Engineering, Universiti Malaysia Pahang, Lebuhraya Tun Razak, 26300 Kuantan, Pahang Darul Maknur, Malaysia
[b] School of Computer Science, University of Nottingham Malaysia Campus, Jalan Broga, 43500 Semenyih, Selangor Darul Ehsan, Malaysia
[c] Department of Computer Science, Faculty of Electrical Engineering, Czech Technical University Karlovo n'am 13, 121 35 Praha 2, Czechia

ARTICLE INFO

Article history:
Received 21 July 2016
Revised 1 March 2017
Accepted 3 March 2017
Available online 6 March 2017

Keywords:
Software testing
t-way testing
Hyper-heuristics
Meta-heuristics
Fuzzy Inference Selection

ABSTRACT

Recently, many meta-heuristic algorithms have been proposed to serve as the basis of a *t*-way test generation strategy (where *t* indicates the interaction strength) including Genetic Algorithms (GA), Ant Colony Optimization (ACO), Simulated Annealing (SA), Cuckoo Search (CS), Particle Swarm Optimization (PSO), and Harmony Search (HS). Although useful, meta-heuristic algorithms that make up these strategies often require specific domain knowledge in order to allow effective tuning before good quality solutions can be obtained. Hyper-heuristics provide an alternative methodology to meta-heuristics which permit adaptive selection and/or generation of meta-heuristics automatically during the search process. This paper describes our experience with four hyper-heuristic selection and acceptance mechanisms namely Exponential Monte Carlo with counter (EMCQ), Choice Function (CF), Improvement Selection Rules (ISR), and newly developed Fuzzy Inference Selection (FIS), using the *t*-way test generation problem as a case study. Based on the experimental results, we offer insights on why each strategy differs in terms of its performance.

1. Introduction

Testing is an important process in software development to help identify areas where the software is not performing as expected. This process is often expensive owing to the time taken to execute the set of test cases. It has been a research focus to find suitable sampling strategies to generate a small yet efficient set of test cases for testing a software system.

Over the years, a plethora of sampling strategies has been proposed in the literature (including that of boundary value analysis, equivalence partitioning, and decision tables; to name just a few). Although useful for some classes of software testing problems, these sampling strategies have not been designed to effectively deal with faults due to interaction. For this reason, many (sampling) *t*-way strategies (where *t* indicates the interaction strength) have been proposed in the scientific literature. Some early algebraic based *t*-way strategies exploit exact mathematical properties of orthogonal arrays. These *t*-way strategies are often fast and produce optimal solutions, yet, they impose restrictions on the supported configurations and interaction strength. The emergence of computational based *t*-way strategies ease these restrictions allowing for

* Corresponding author.
E-mail addresses: kamalz@ump.edu.my (K.Z. Zamli), fakhruddin@uom.edu.pk (F. Din), Graham.Kendall@nottingham.edu.my (G. Kendall), albeybes@fel.cvut.cz (B.S. Ahmed).

http://dx.doi.org/10.1016/j.ins.2017.03.007
0020-0255/

Third Q1 Paper:

PLOS ONE

RESEARCH ARTICLE

A hybrid Q-learning sine-cosine-based strategy for addressing the combinatorial test suite minimization problem

Kamal Z. Zamli[1]*, Fakhrud Din[1], Bestoun S. Ahmed[2], Miroslav Bures[2]

1 IBM Centre of Excellence, Faculty of Computer Systems and Software Engineering, Universiti Malaysia Pahang Lebuhraya Tun Razak, 26300 Kuantan, Pahang Darul Makmur, Malaysia, 2 Software Testing Intelligent Lab (STILL), Department of Computer Science, Faculty of Electrical Engineering Czech Technical University, Karlovo nam. 13, 121 35 Praha 2, Czech Republic

* kamalz@ump.edu.my

OPEN ACCESS

Citation: Zamli KZ, Din F, Ahmed BS, Bures M (2018) A hybrid Q-learning sine-cosine-based strategy for addressing the combinatorial test suite minimization problem. PLoS ONE 13(5): e0195675. https://doi.org/10.1371/journal.pone.0195675

Editor: Peter R. Lewis, Aston University, UNITED KINGDOM

Received: March 27, 2017

Accepted: March 27, 2018

Published: May 17, 2018

Data Availability Statement: All data underlying the study are within the paper.

Funding: This study is funded by the Science Fund Grant for the project entitled, "Constraints T-Way Testing Strategy with Modified Condition/Decision Coverage" from the Ministry of Science, Technology, and Innovation, Malaysia. The work reported in this paper is funded by Fundamental Research Grant from Ministry of Higher Education Malaysia titled: A Reinforcement Learning Sine Cosine based Strategy for Combinatorial Test Suite

Abstract

The sine-cosine algorithm (SCA) is a new population-based meta-heuristic algorithm. In addition to exploiting sine and cosine functions to perform local and global searches (hence the name sine-cosine), the SCA introduces several random and adaptive parameters to facilitate the search process. Although it shows promising results, the search process of the SCA is vulnerable to local minima/maxima due to the adoption of a fixed switch probability and the bounded magnitude of the sine and cosine functions (from -1 to 1). In this paper, we propose a new hybrid Q-learning sine-cosine- based strategy, called the Q-learning sine-cosine algorithm (QLSCA). Within the QLSCA, we eliminate the switching probability. Instead, we rely on the Q-learning algorithm (based on the penalty and reward mechanism) to dynamically identify the best operation during runtime. Additionally, we integrate two new operations (Lévy flight motion and crossover) into the QLSCA to facilitate jumping out of local minima/maxima and enhance the solution diversity. To assess its performance, we adopt the QLSCA for the combinatorial test suite minimization problem. Experimental results reveal that the QLSCA is statistically superior with regard to test suite size reduction compared to recent state-of-the-art strategies, including the original SCA, the particle swarm test generator (PSTG), adaptive particle swarm optimization (APSO) and the cuckoo search strategy (CS) at the 95% confidence level. However, concerning the comparison with discrete particle swarm optimization (DPSO), there is no significant difference in performance at the 95% confidence level. On a positive note, the QLSCA statistically outperforms the DPSO in certain configurations at the 90% confidence level.

Introduction

An optimization problem relates to the process of finding the optimal values for the parameters of a given system from all possible values with minimum or maximum profitability. In past decades, many meta-heuristic algorithms have been proposed in the scientific literature