A HYBRID ADAPTIVE HARMONY SEARCH WITH MODIFIED GREAT DELUGE ALGORITHM FOR SCHOOL TIMETABLING

BILLEL ARBAOUI

DOCTOR OF PHILOSOPHY
UNIVERSITI UTARA MALAYSIA
2025

Awang Had Salleh
Graduate School
of Arts And Sciences

Universiti Utara Malaysia

PERAKUAN KERJA TESIS / DISERTASI
(Certification of thesis / dissertation)

Kami, yang bertandatangan, memperakukan bahawa
(We, the undersigned, certify that)

ARBAOUI BILLEL

calon untuk Ijazah **PhD**
(candidate for the degree of)

telah mengemukakan tesis / disertasi yang bertajuk:
(has presented his/her thesis / dissertation of the following title):

"A HYBRID ADAPTIVE HARMONY SEARCH WITH MODIFIED GREAT DELUGE ALGORITHM FOR SCHOOL TIMETABLING"

seperti yang tercatat di muka surat tajuk dan kulit tesis / disertasi.
(as it appears on the title page and front cover of the thesis / dissertation).

Bahawa tesis/disertasi tersebut boleh diterima dari segi bentuk serta kandungan dan meliputi bidang ilmu dengan memuaskan, sebagaimana yang ditunjukkan oleh calon dalam ujian lisan yang diadakan pada : ***28 Mac 2024.***
That the said thesis/dissertation is acceptable in form and content and displays a satisfactory knowledge of the field of study as demonstrated by the candidate through an oral examination held on: ***28 March 2024.***

Pengerusi Viva: *(Chairman for VIVA)*	**Assoc. Prof. Ts. Dr. Maslinda Mohd Nadzir**	Tandatangan *(Signature)*
Pemeriksa Luar: *(External Examiner)*	**Assoc. Prof. Dr. Mohd Nizam Mohmad Kahar**	Tandatangan *(Signature)*
Pemeriksa Dalam: *(Internal Examiner)*	**Assoc. Prof. Dr. Yuhanis Yusof**	Tandatangan *(Signature)*
Nama Penyelia/Penyelia-penyelia: *(Name of Supervisor/Supervisors)*	**Dr. Juliana Wahid**	Tandatangan *(Signature)*
Nama Penyelia/Penyelia-penyelia: *(Name of Supervisor/Supervisors)*	**Assoc. Prof. Dr. Syariza Abdul Rahman**	Tandatangan *(Signature)*

Tarikh:
(Date) ***28 March 2024***

Permission to Use

I agree that the Universiti Library may make this thesis publicly available for inspection as a condition of my presenting it in fulfillment of the requirements for a postgraduate degree from Universiti Utara Malaysia. I also acknowledge that, in the event of their absence, my supervisor(s) or the dean of the Awang Had Salleh Graduate School of Arts and Sciences may grant permission for this thesis to be copied, in whole or in part, for scholarly purposes. It is understood that without my prior consent, no part of this thesis may be copied, published, or utilized for commercial gain. Furthermore, it is expected that both Universiti Utara Malaysia and I will receive proper acknowledgment for any scholarly use made of any part or the entirety of my thesis.

Please contact the following if you would want permission to copy or otherwise use any of the contents in this thesis, in whole or in part:

Dean of Awang Had Salleh Graduate School of Arts and Sciences
UUM College of Arts and Sciences
Universiti Utara Malaysia
06010 UUM Sintok

Abstrak

Masalah Penjadualan Sekolah Menengah (HSTP) adalah masalah NP-lengkap yang melibatkan penjanaan jadual mingguan tanpa konflik dalam penugasan guru dan slot waktu. Algoritma Pencarian Harmoni (HSA) merupakan metaheuristik berkesan dalam menyelesaikan HSTP kerana kecekapannya serta keperluan penyetelan parameter yang minimum. Namun, kajian terdahulu sering mengabaikan interaksi antara parameter yang mengawal keseimbangan eksplorasi dan eksploitasi, menyebabkan kesukaran melarikan diri daripada optima tempatan serta menjejaskan kualiti penyelesaian. Kajian ini menambah baik HSA dengan menggabungkannya bersama Algoritma Great Deluge (GDA) melalui metodologi empat fasa. Fasa 1 memodelkan peraturan heuristik dalam peringkat penugasan masa bagi meningkatkan algoritma pembinaan sedia ada, Fasa 2 melaraskan parameter secara adaptif berdasarkan kedudukan lelaran, bilangan penyelesaian, status tingkah laku serta hubungan antara parameter, manakala Fasa 3 meningkatkan kepelbagaian gerakan kejiranan dalam ruang pencarian melalui penggabungan HSA dan GDA, dan Fasa 4 menilai algoritma menggunakan set data penanda aras serta kes dunia sebenar. Hasil pengiraan menunjukkan bahawa model pembinaan yang diperbaiki dapat menghasilkan penyelesaian lebih baik, di mana HSA adaptif memperoleh penyelesaian terbaik bagi 12.8% kes (5/39) dan mencatat keputusan setara dalam 10.3% kes (4/39), sementara algoritma gabungan mencapai penyelesaian terbaik dalam 10.3% kes (4/39) serta setara dalam 10.3% kes (4/39) berbanding kaedah terkini. Kesimpulannya, algoritma yang dicadangkan terbukti berkesan dalam menyelesaikan HSTP, menjadikannya strategi ampuh untuk menyelesaikan masalah pengoptimuman kompleks, khususnya dalam penjadualan sekolah menengah.

Kata kunci: Pendekatan Adaptif, Algoritma Great Deluge, Algoritma Pencarian Harmoni, Hibridisasi, Penjadualan Waktu Sekolah.

Abstract

High school timetabling problem (HSTP) is an important NP-complete problem to generate a weekly-based timetable for classes, avoiding conflict of teachers and timeslots, which has been actively researched spanning many decades to this day. Harmony Search Algorithm (HSA) is one superior metaheuristic method to solve timetabling problem due to its search efficiency and less parameters settings. However, previous studies often overlooked some crucial factors on the interaction among parameters that control the balance between exploration and exploitation during the search process. Due to the unbalance exploration and exploitation, the search unable to jump out of local optima that deteriorate the solution quality. This research enhances HSA by hybridizing it with the Great Deluge Algorithm (GDA) using a four-phase methodology. In Phase 1, heuristic rules refine the time assignment stage to improve an existing construction algorithm. Phase 2 adaptively tunes parameters based on iteration position, solution number, behavioral status, and parameter linkages. Phase 3 enhances search diversity by integrating HSA and GDA, and Phase 4 evaluates the algorithms on benchmark and real-world datasets. Computational results demonstrate that the improved constructive model generates better initial solutions. The adaptive HSA and hybrid HSA-GDA further enhance solution quality, with the adaptive HSA achieving best-known solutions for 12.8% of instances (5/39) and tying in 10.3% (4/39), while the hybrid approach attains best-known solutions in 10.3% (4/39) and ties in 10.3% (4/39) against state-of-the-art methods. In conclusion, the proposed algorithms significantly improve HSTP solutions, demonstrating their superiority and establishing them as effective strategies for solving complex optimization problems, particularly in high school timetabling.

Keywords: Adaptive Approach, Great Deluge Algorithm, Harmony Search Algorithm, Hybridization, School timetabling**.**

Acknowledgement

First and foremost, I express my deepest gratitude to ALLAH, the Most Gracious and the Most Merciful, for bestowing upon me the strength, guidance, and determination to undertake this research journey.

I owe a profound debt of gratitude to my parents and my loving wife. Their unwavering love, support, and sacrifices have been the backbone of my academic pursuits. My parents have always illuminated my path with their encouragement and prayers, and my wife, with her patience, understanding, and companionship, has been a constant source of strength throughout this journey. I am forever thankful for their collective belief in me.

I am immensely grateful to my two esteemed supervisors, Prof. Juliana binti Wahid and Prof. Madya Dr. Syariza Binti Abdul Rahman. Their invaluable insights, persistent guidance, and constructive feedback have been pivotal to the success of this research. Their mentorship has not only shaped this thesis but has also profoundly influenced my academic and personal growth.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

BGDA	Basic Great Deluge Algorithm
DHSA	Adaptive Harmony Search Algorithm
GDA	Great Deluge Algorithm
HSA	Harmony Search Algorithm
HSTP	High School Timetabling Problem
KHE	Kingston High School Timetabling Engine
MCHR	Markov Chain Heuristic Rules Model
MGDA	Modified Great Deluge Algorithm
NS	Neighbourhood Structure
NSs	Neighbourhood Structures
XHSTT	Extensible Markup Language for High School Timetabling

CHAPTER ONE

INTRODUCTION

1.1 Background of the Study

Timetabling problem defines the problem of allocating given resources to objects at a certain time and space, subject to constraints, in such a way that the allocation fulfills a group of suitable objectives as much as possible (Gashgari et al., 2021; Wren, 1996). This problem appears in various areas, such as sport timetabling (Silva et al., 2020), vehicle timetabling (Schiewe, 2020), employee timetabling (Saraswati et al., 2021) and educational timetabling (Alencar et al., 2020; Gashgari et al., 2021; Tan et al., 2021).

Educational timetabling involves school, examination, and course timetabling. These three educational timetabling areas have two similarities. The first is the general processing of optimization methodologies, of which there are four processes, namely, formulation of problem (named dataset), construction, optimization, and evaluation (Gashgari et al., 2021). The second similarity is the assignment or movement of educational timetabling in the construction and optimization phases, though there are differences in the constraints of each domain. For instance, time or resources are assigned to a meeting, event, course, or exam for the construction phase, and neighborhood structures (NSs) are swapped or moved according to the time or resources for the optimization phase (Tan et al., 2021).

Generally, many problems of educational timetabling variations have been propositioned in the literature based only on two facts. The first fact is the parameters of the school timetabling problem are more complex which require sophisticated

optimization techniques and algorithms to create schedules that meet all constraints and preferences while maximizing efficiency and resource utilization e.g., i) resource: students, teachers, classes or rooms; ii) event: meets, times or slots compared to course and examination timetabling problem parameters (e.g., time or slot) (Gashgari et al., 2021). The second fact that school timetabling problem is significantly influenced by the individual policies and structures of educational institutes, primarily focusing on creating a weekly schedule that prevents teachers from being assigned to two classes at the same time. In contrast, the complexities of university course and examination timetabling are largely dependent on the total number of students in the institution, necessitating careful planning to avoid overlap of lectures and exams for courses attended by common students, thereby fostering a more organized and efficient academic schedule (Bashab et al., 2020; Ceschia et al., 2023; Gashgari et al., 2021; Pillay, 2014). From these two facts, the current study focuses on timetabling problem research which is the school timetabling problem.

Solving the High School Timetabling Problem (HSTP) is crucial because it directly impacts the efficiency and effectiveness of educational institutions (Moral et al., 2023). An optimized timetabling system ensures that resources such as classrooms and teachers are used most effectively, leading to better educational outcomes. Furthermore, a poorly managed timetable can result in conflicts, underutilization of resources, and increased operational costs, which can significantly hinder the educational process (Goode et al., 2024). Given the complex and dynamic nature of educational environments, developing robust solutions for HSTP can lead to substantial improvements in administrative efficiency and educational quality (Liu & Yu, 2023).

The goal of researchers is to find an effective algorithm for finding the optimal solution to the high school timetabling problem (HSTP). This has led them to propose a formulation of HSTP as an optimization method for minimizing the constraint violations (Schaerf, 1999). HSTP research has been investigated since 1964 (Berghuis et al., 1964). Researchers have been studying the use of a standardized data format or language to indicate school timetabling problems and, as a consequence, facilitate the analysis and exchange of problems among researchers (Pillay, 2014). The data format includes OR-Library (Smith et al., 2003) and Extensible Markup Language for High School Timetabling (XHSTT) (Post et al., 2012).

Although enormous efforts have been applied over the last 15 years to discover efficient methods and approaches to solve HSTP, the solution algorithms are typically problem-specific, which means that no general solution can solve various HSTP instances. There is still a non-particular method that can offer superb solution performance beyond the broad scale of HSTP instances. This coincides with the no-free-lunch theorem (Wolpert & Macready, 1997) that says no strategy can be expected to perform better than any other. Indeed, HSTP is categorized as a nondeterministic polynomial complete problem, which indicates that existing methods and algorithms are unable to discover an optimal solution for certain instances of the problem (Even et al., 1976). The objective function of HSTP mainly consists of hard and soft constraints. For example, XHSTT (one of HSTP dataset) contains 16 constraints indicate desired times for events, unavailable hours for resources, and other criteria. Based on dataset, each constraint has the option to be marked as mandatory (hard constraint) or optional (soft constraint) (Kingston, 2014). In HSTP, a critical aspect to consider is the differentiation between "hard" and "soft" constraints. Hard constraints are non-negotiable requirements that

must be strictly adhered to in order to construct a feasible timetable. These typically include avoiding scheduling conflicts where a teacher is assigned to two classes at the same time and ensuring that rooms are not double-booked. While soft constraints are not obligatory, adhering to them can significantly enhance the efficiency and functionality of HSTP. These might involve efforts to minimize cases where the number of students exceeds the room capacity or trying to schedule all teachers of classes in the same room (Ceschia et al., 2023).

The solution algorithms for HSTP can be classified into construction and improvement algorithms (Tan al., 2021). The construction algorithm of a timetable for HSTP always builds solution components from an empty set until it finds one or group candidate solution that fulfills the requirement of assigning resources (teachers, times, rooms, and students) to lesson groups while minimizing the constraint violations (Tans 2021). Some construction algorithms for HSTP include Kingston High School Timetabling Engine (KHE) (Kingston, 2014), genetic algorithm (GA) (Filho & Lorena, 2001), THOR (Tabelas Horárias, or THOR in Portuguese, is a word that approximately translates to timetabling charts) (Melício et al., 2006), and greedy randomized constructive procedure (GRASP) (Santos et al., 2005). Three theoretical factors support KHE as a unique construction algorithm compared to the other three (GA, THOR, and GRASP). Firstly, KHE generates high-quality solutions efficiently (Ceschia et al., 2022). Secondly, it does not require extensive searching, making it robust for various optimization problems (Ceschia et al., 2022). Thirdly, KHE is capable of handling datasets of different sizes, utilizing XHSTT datasets (Kingston, 2024; Post et al., 2012; Tan et al., 2021). Based on these facts, this study adopted and enhanced the KHE for solving the HSTP in the construction phase.

In the context of improvement, the optimization algorithm for solving the HSTP is typically in an iterative manner, and thus multiple evaluations of objective functions are required. Different algorithms may have different effectiveness and constraints (Yang, 2021). The optimization techniques are aimed to find the improvement of HSTP candidate solutions that are better than the solutions produced by the construction methods (Tan et al., 2021). Numerous state-of-the-art optimization techniques were suggested to solve HSTP includes the exact methods, metaheuristic and hyper-heuristic algorithms that can find near optimal or optimal solutions (Pillay, 2014). Integer programming technique is a mathematical model similar to the cuts and branching techniques that are always looking for an optimal or lower-bound solution of various data instances (Fonseca et al., 2017). A metaheuristic is either a lower-level or higher-level procedure designed to create a suitable good solution to an optimization problem (Du & Swamy, 2016). Hyper-heuristic is related to heuristic area which can explore or choose other heuristic algorithms (Du & Swamy, 2016). Metaheuristic algorithms are chosen for the thesis due to their flexibility, robustness, and efficiency in tackling a wide range of optimization problems. Metaheuristic algorithms balance exploration and exploitation effectively, making them suitable for solving complex, large-scale issues where traditional methods may fail (Cuevas et al., 2021; Cuevas, Luque, et al., 2024; Hussain et al., 2019). Inspired by natural processes, such as evolution, social behavior, and foraging, these algorithms offer diverse strategies and have a proven track record of success in similar problems. Furthermore, their ability to be customized and hybridized enhances their performance, and computational experiments can provide empirical evidence of their efficacy (Martí et al., 2024), justifying their selection for this thesis.

Exploration and exploitation are fundamental concepts in metaheuristic algorithms. Exploration refers to the ability of an algorithm to investigate diverse areas of the search space to discover global optima, while exploitation focuses on intensively searching around known good solutions to refine them further. A balanced approach between exploration and exploitation is crucial for achieving high-quality solutions efficiently (Brahim et al., 2024). Overemphasis on exploration can lead to excessive search times without finding optimal solutions, whereas too much exploitation can result in premature convergence to suboptimal solutions. Therefore, effective algorithms must strike a balance between these two strategies to ensure a thorough yet efficient search process (Molina et al., 2020).

Metaheuristic algorithms are chosen for this study due to their proven ability to handle complex optimization problems that are characterized by non-linearity, large solution spaces, and multiple local optima (Jiang & Fan, 2023). Unlike exact optimization methods, which often become computationally infeasible for large-scale problems, metaheuristics provide a balance between exploration (searching globally across the solution space) and exploitation (refining locally optimal solutions) (Zhang et al., 2024). This makes them well-suited for finding near-optimal solutions within a reasonable computation time when exact solutions are impractical (Jiang & Fan, 2023; Stein & Bäck, 2024; Yang, 2024; Zhang et al., 2024). Furthermore, metaheuristic algorithms are flexible and can be adapted to various problem structures without requiring gradient information, making them particularly advantageous in scenarios where the objective function is complex, discontinuous, or non-differentiable (Cuevas, Zaldívar, et al., 2024). Furthermore, this study focuses on metaheuristic algorithms which consists of local-search and population algorithms (Blum & Roli, 2003). Local

search-based algorithm explores and focuses on the area around the current solution (Koopialipoor & Noorbakhsh, 2020a). Examples of such algorithms are simulated annealing (SA) (Assad & Deep, 2018) and great deluge algorithm (GDA) (Eng et al., 2020). Population-based algorithm is a set of solutions that are iteratively updated until the termination requirement is satisfied (Du & Swamy, 2016). Examples of such algorithms are GA (Reddy et al., 2020), particle swarm optimization (PSO) (Wang et al., 2021), and harmony search algorithm (HSA) (Abualigah et al., 2020).

Established by Geem et al. (2001), HSA is inspired by the improvisation method of jazz musicians. HSA has the ability to diversify (explore) through population solution while requiring less exploitation of the solution (Koopialipoor & Noorbakhsh, 2020a). HSA has been utilized to numerous science and engineering domain area such as scheduling (Abu Doush et al., 2022), electrical engineering (Riquelme-Dominguez et al., 2022), healthcare (Abdulkhaleq et al., 2022), and mechanical engineering problems (Uray et al., 2022). The justification for using HSA in this thesis is rooted in its unique combination of strengths, which align closely with the requirements of solving the given problem. HSA's simplicity and ease of implementation make it a practical choice (Askarzadeh & Rashedi, 2017), especially for problems requiring rapid adaptation and minimal computational overhead. Its inherent ability to balance exploration and exploitation ensures a diverse search through the solution space while refining promising solutions (Wang et al., 2023), which is crucial for addressing the complexity of the HSTP. Furthermore, HSA's proven versatility across various domains—such as scheduling, engineering, and healthcare—demonstrates its robustness and adaptability in tackling optimization problems with varying constraints and objectives (Madhavi et al., 2024). Additionally, HSA's potential for hybridization with other algorithms further

strengthens its justification, as combining it with complementary techniques enhances its performance and applicability. For instance, hybrid frameworks, such as Opposition-Based Harmony Search (OBHS) integrated with Manta Ray Foraging Optimization (MRFO) (Somashekar & Jagirdar, 2024), have demonstrated significant improvements in addressing complex optimization challenges, making HSA a flexible and powerful choice for this study. From this perspective, this study selects HSA to further investigate modifications to its design structure for solving HSTP.

In the context of HSA, the parameters HMCR (harmony memory consideration rate), PAR (pitch adjustment rate), and RCR (random consideration rate) play significant roles in balancing exploration and exploitation. HMCR determines the probability of choosing solutions from the harmony memory, encouraging exploitation by refining existing solutions. PAR controls the likelihood of adjusting a solution, introducing variability and aiding exploration. RCR influences the probability of generating entirely new solutions, enhancing the exploration of the search space. The proper tuning of these parameters is essential to maintain an optimal balance between exploration and exploitation, thereby improving the overall effectiveness of the HSA. Consequently, modifications the HSA have become more popular among researchers to overcome these inherent weaknesses. For instance, modified HSAs are often preferred (Chen et al., 2012; Hasanipanah et al., 2020; Li et al., 2017; Mahdavi et al., 2007; Peraza et al., 2016; Wang & Huang, 2010; Wang et al., 2019; Hasanipanah et al., 2022; Quan et al., 2021), as they can dynamically tune parameters to better adapt to complex problems.

Several hybridized variants of HSA have been introduced in the literature (Du & Swamy, 2016). This includes the hybridization of HSA with hill climbing for the nurse

rostering problem (Awadallah et al., 2017), hybridization of HSA with Nelder–Mead algorithm for the combined heat and power economic dispatch problem (Feng et al., 2017), hybridization of HSA with SA for HSTP (Shambour et al., 2013), and hybridization of HSA with GDA for course timetabling problem (Wahid & Hussin, 2017; Wahid, 2017; Wahid et al., 2018, 2019). Even though the work of Shambour et al. (2013) solved the same domain of problem with this study, their works implemented the hybridization of HSA with SA in which the SA method was employed outside the HSA procedure as a technique of polishing aimed to increase the most successful outcome currently found. However, the only existing work by Shambour et al. (2013) focused on only a few datasets, the majority of which are small. Additionally, the SA is incompatible with HSTP, as HSTP environments are often larger and more complex, meaning their HSA has several limitations in handling exploitation. The basic HSA faces several limitations that necessitate its hybridization with other algorithms. It struggles with regulating and adjusting parameters, which impacts the quality of the solutions obtained (Brambila-Hernández et al., 2023). The fixed parameters in HSA make it less adaptable, leading to poor performance in complex optimization problems. Furthermore, HSA often suffers from premature convergence, getting stuck in local optima, and exhibits slow convergence speed, making it inefficient in reaching optimal solutions. Hybridization with other algorithms assists in escaping local optima and improves overall performance. HSA is well-suited for hybridization due to its ability to balance exploration and exploitation through its improvisation process and memory-based structure, enabling it to explore diverse solutions while retaining high-quality candidates (Dubey et al., 2021; Wang et al., 2023). GDA was selected due to its capability of accepting worse solutions to avoid stagnation and improve solution quality, as well as its parameter-driven water level acceptance mechanism that can be

tailored to problem-specific objectives, offering greater adaptability and precision (Acan & Ünveren, 2015). From this point of view, this study intends to explore GDA to be hybridized inside the HSA procedure.

GDA was proposed by Dueck (Dueck, 1993). It is a local search-based metaheuristic for continuous global optimization that allows inferior solutions using the solution search approach based on an acceptance rule. This assists in the escape from local optima (Du & Swamy, 2016). Furthermore, numerous methods were hybridized with GDA in the literature to improve the intensification of GDA. These involves hybridization of GDA with variable neighborhood search algorithm for efficient task scheduling in the field of grid computing (Eng et al., 2020), hybridization of GDA with GA for rough set attribute reduction (Jaddi & Abdullah, 2013), and hybridization of GDA with tabu search for quadratic assignment problem (Acan & Ünveren, 2015).

The GDA starts with an initial solution that contains the objective function value and this value is reduced by a particular rate (water level or threshold accepting) at each iteration (Wahid & Hussin, 2017). This type of GDA is called linear which contains water level as its parameter. Another type of GDA is non-linear (Guha et al., 2021). The non-linear GDA contains two extra parameters namely minimum and maximum rate which will be set manually depending on the size of datasets. This study focuses on linear type as this type of GDA uses fewer parameters.

1.2 Problem Statement

The KHE algorithm consists of four stages: structure, time assignment, resource assignment, and clean-up. In the time assignment stage, layers are arranged using these

rules: 1) Arrange assigned layers first; 2) If both layers are unassigned, prioritize by duration; 3) If durations are equal, prioritize by demand; 4) If both duration and demand are equal, sort by index (Kingston, 2021). However, the problem with the time assignment stage of KHE algorithm is that it does not integrate duration, demand, and index values into a single, cohesive metric for layer difficulty. This leads to increased assignment costs since the heuristic rules fail to capture the actual complexity of the layers accurately. These heuristic rules fail to fulfill the minimum cost of matching the meets and times that affect the objective function utilizing XHSTT datasets (Kingston, 2014, 2024) since the decision to choose the most difficult layer based on duration, demand, and index values has been separated. There are so far no strategies have been developed in the literature to combine all those information in assessing the difficulty of a layer. For instance, if layer A (duration: 10, demand: 5, and index: 10) and layer B (duration: 9, demand: 15, and index: 16), then sorting based on the heuristic rules will first choose the layer A to assign the time, as the duration value is higher than layer B. However, layer B contains more demand values compared to the layer A which make it more difficult. Therefore, sorting procedures that combine all these values are needed to enhance the time assignment stage in the KHE algorithm, thus minimizing the assignment cost and producing a quality solution.

Moving forward, the initial solution of HSTP can be improved by HSA because it is known as one of the superior approaches for solving timetabling problems (Anwar et al., 2013; Awadallah et al., 2017; Szwarc & Boryczka, 2022). The HSA requires several parameters settings, including harmony memory consideration rate (HMCR), random consideration rate (RCR), pitch adjustment rate (PAR), and iteration size. However, research on the parameter tuning of HSA within the timetabling problem is still lacking

because most studies considered fixed values of the parameters in the HSA (Al-Betar & Khader, 2012; Shambour et al., 2013; Wahid & Hussin, 2014). Studies have shown that fixing the parameter values of HSA has some limitations. This condition limits the search to adjust the solution accordingly during the search process and converge quickly (Mahdavi et al., 2007a). The fixed parameters have no relation between search space and problem, which lead the search space to isolate from the problem and non-good results achieved towards the end of the iterations (Wang et al., 2019).

Several researchers have shown various improvements to HSA in adjusting the parameters dynamically to stabilize the impact of control parameters of the search space and produce promising solutions (Chen et al., 2012; Hasanipanah et al., 2020; Li et al., 2017; Mahdavi et al., 2007; Peraza et al., 2016; Wang & Huang, 2010; Wang et al., 2019; Hasanipanah et al., 2022; Quan et al., 2021). However, there remains a deficiency in research on evolutionary strategies within adaptive HSA aimed at balancing search exploration and exploitation. The primary issues with current adaptive HSA can be framed in terms of key optimization concepts: limited search space, memory, and decision-making processes. First, the constrained search space in previous studies is a critical limitation that must be addressed, as it has not highlighted the importance of considering additional dimensions and aspects within the search process to resolve this issue effectively, thereby necessitating the investigation of broader dimensions in the search space (Plaat, 2024). Second, previous studies lack an effective memory component for the search space, highlighting the need for a comprehensive analysis of the search space behavior (Schneider et al., 2023). Third, the decision-making processes within the search algorithms (such as pitch adjustment and random consideration rates) used in previous studies are limited and require thorough examination and improvement

(Tsirtsis et al., 2024). Parameter setting is critical in optimization algorithms like HSA because it significantly affects the algorithm's performance and the quality of the solutions obtained. Proper parameter tuning can enhance the algorithm's ability to explore the search space effectively and avoid premature convergence to suboptimal solutions. Fixed parameters can limit the flexibility and adaptability of the algorithm, leading to inefficiencies and poor performance. Dynamic parameter tuning, on the other hand, allows the algorithm to adjust its behavior based on the search process's current state, leading to better solutions and more efficient search processes. The implications of inadequate parameter settings include prolonged search times, increased computational costs, and suboptimal solutions that fail to meet the problem's constraints and objectives (Ma et al., 2023).

Several researchers have demonstrated the hybridization of HSA with local search methods, such as course timetabling (Wahid & Hussin, 2017; Wahid, 2017; Wahid et al. 2018, 2019) nurse rostering (Awadallah et al., 2017), economic dispatch problem (Feng et al., 2017). However, existing hybrid HSAs have several limitations. Firstly, the process for selecting the local search algorithm to hybridize with HSA is not using an efficient search process algorithm, making it important to explore more decent search methods (Roth & Corsi, 2023; Silveira et al., 2020). Secondly, the integration and tuning of hybrid parameters between the two algorithms are being locked in local optimums, which highlights the need for improved optimization strategies to unlock better global solutions and enhance overall performance (Koopialipoor & Noorbakhsh, 2020b). Thirdly, current hybrid HSAs exhibit gaps that did not have proper trade-offs between exploration and exploitation in the search space, emphasizing the need for refined techniques to achieve a more optimal balance and improve search efficiency

(Abualigah et al., 2020). To address these issues, this research proposes integrating GDA, a local search-based algorithm with fewer parameters, into the hybrid HSA framework. This approach will focus on developing efficient algorithm selection strategies, advanced parameter tuning methods to avoid local optima, and refined techniques to achieve a better balance between exploration and exploitation, ultimately enhancing the overall performance and efficiency of hybrid HSA.

Several studies have improved water level or threshold accepting of GDA to solve the timetabling problem through modification of the water level rate (Burke et al., 2006; Kheiri & Keedwell, 2017; Landa-Silva & Obit, 2008a; Mushi, 2011). GDA accepts inferior solutions that near to the best solution based on water level acceptance rule that contains the decay rate. However, the non-linearity of water level acceptance with cost function is still lack in previous studies. Thus, the feasibility of solution and type of constraints are still separated from the decay rate. This limits the solution improvement since it fails to fully meet some of the hard and soft constraints' requirements (Innocente & Sienz, 2021; Mallipeddi et al., 2015; Rahimi et al., 2023). Based on the previous works on hybridization HSA with GDA (Al-Betar et al., 2013; Choudhuri & Ravi, 2010; Wahid & Hussin, 2017), the GDA contributes for exploitation, whereas HSA works for exploration. Moreover, in the studies, the GDA water level or threshold accepting rules are only used during the local search space since the behavioral neighborhood moves of GDA are unhandled by HSA. Nevertheless, the decay water level rate should be based on these constraints in order to synchronize the linearity of decay water level rate. This study chooses GDA due to two reasons. Firstly, GDA accepts worse solutions to increase the chances of finding the best one and avoid stagnation, unlike HC. Secondly, it also uses a water level rate tied to the objective

function, while SA relies on probability, which can hinder performance. In addition, this study hybridized adaptive HSA with GDA and modify the water level in GDA to give new possibility paths during the local search space related to the feasibility of the best solution obtained (objective function).

1.3 Research Questions

Based on the aforementioned sections, numerous research questions motivated this research are as follows:

1. How to improve the heuristic strategy for layer sorting in the KHE algorithm's time assignment stage, and are the resulting algorithms capable of handling and solving HSTP?
2. How to improve the existing HSA that adaptively changes the parameter settings, which demonstrate efficacy on solving HSTP?
3. How to improve adaptive HSA by considering hybridization with local search, and how effectively these enhancements handle and solve HSTP?
4. How effective can real-world datasets of HSTP be evaluated and validated through the proposed algorithms?

1.4 Research Objectives

The goal of this thesis is to hybridize HSA with GDA algorithms for solving the HSTP using XHSTT data instances. To achieve the main goal, the study involves the following individual objectives:

1. To model heuristic rules in the time assignment stage to address the inefficient layer arrangement in the KHE construction algorithm, thereby enhancing its overall performance.

2. To improve the parameter setting process in HSA by addressing inefficiencies caused by static parameter settings. This will be achieved by formulating an adaptive function that recalculating parameters which consider iteration position, solution number, bahavioral status and parameter linkages.
3. To hybridize the adaptive HSA with two decay rates of GDA to solve the problem of limited search space exploration, thereby providing more possibilities for both exploration and exploitation.

1.5 Scope of the Study

The overall scope of the thesis is on the use of a metaheuristic population-based algorithm so-called HSA and a metaheuristic local search algorithm such as GDA to solve and optimize the HSTP. The proposed approaches will be tested on well-known benchmark datasets of HSTP[1] and real-world dataset of HSTP of Malaysia case. The benchmark datasets of HSTP, contributed by 13 countries, follows an XX-YY-ZZ naming pattern, representing country, institution, and year. It includes 39 datasets categorized by size. The smallest is the "VillageSchool" from the Czech Republic, and the largest is "GEPRO" from the Netherlands.

It is important to note the limitation of this research in which the evolutionary approaches that do not fall within this context are not considered in this study. The focus is solely on the evolutionary approaches within the HSA framework.

[1] High School Timetabling Project (https://www.utwente.nl/en/eemcs/dmmp/hstt/).

1.6 Significance of the Research

The main achievements, including contributions to the field of metaheuristic, can be beneficial to the development of hybridizing HSA with GDA using well-known XHSTT data instances across the world. From this research, the scientific contributions are:

1. This research introduces a heuristic-based model for sorting layers during the time assignment stage in the KHE construction algorithm. This theoretical advancement provides new directions for academic exploration in optimization techniques, offering fresh insights into the design of metaheuristic algorithms
2. A model that re-estimates the parameters (HMCR, PAR, and RCR) rates of HSA at each iteration. The parameters will be adaptively changed based on the priority of several neighborhood structures in HSA, adding a new dimension to how researchers can optimize and fine-tune algorithms dynamically to respond to evolving problem conditions.
3. A hybridization of adaptive HSA with GDA integrates the modification of acceptance rules based on two decay rates of water level, addressing both hard and soft constraints. This emphasizes the adaptive nature of the algorithm, allowing it to dynamically adjust and enhance performance when solving complex optimization problems, thereby increasing its overall robustness and efficiency. Academically, this contribution enriches the metaheuristic landscape by providing an adaptive mechanism that boosts the algorithm's performance in solving complex optimization problems.
4. The proposed construction and improvement algorithms are efficiently solving and optimizing benchmark datasets and real-world datasets of HSTP. These practical contributions demonstrate the algorithm’s capability to efficiently

solve and optimize real-world problems. This directly benefits industries and practitioners in fields such as education, logistics, and resource management, where timetabling and scheduling are crucial.

1.7 Research Dissemination

There are four publications in this study, which one is under review. The publications are as follows:

1. Arbaoui, B., Wahid, J., & Abdul-Rahman, S. (2020). Enhancing the Sorting Layers in the Initial Stage of High School Timetabling. *2020 IEEE 10th Symposium on Computer Applications & Industrial Electronics (ISCAIE)*, pp. 59-63, doi: 10.1109/ISCAIE47305.2020.9108804.
2. B. Arbaoui, J. Wahid, and S. Abdul-Rahman, The School Timetabling Problem at a Malaysian School: A Real-World Datasets and Solutions. *2022 2nd International Conference on Design Innovation, Social Science & Technology 2021 (ICDISST2021),* doi: 10.1063/5.0181827.
3. B. Arbaoui, J. Wahid, and S. Abdul-Rahman, "The school timetabling problem at a Malaysian school: A real-world datasets and solutions," 2024, p. 020083. doi: 10.1063/5.0181827.
4. B. Arbaoui, J. Wahid, S. Abdul-Rahman, E. Özcan, and J. H. Drake, "An Adaptive Evolutionary Approach to High School Timetabling," *Expert Systems with Applications*, 2023. (SCOPUS Q1 UNDER REVIEW).

1.8 Thesis Outline

This thesis consists of eight chapters and the remainder of the chapters are organized as follows. In Chapter Two, the first section provides an introduction to educational

timetabling problems, followed by a focus on the HSTP which is the main topic of this study. The chapter also discusses benchmark datasets and methodologies commonly used in the school timetable communities. The subsequent sections of the chapter present the HSA, Markov Chain theory, and GDA, which are investigated in this study. Chapter Three outlines the approach that will be taken to address the specific research objectives of the study. The chapter starts by introducing a research framework that comprises of three phases, namely construction, improvement, and producing real-world dataset. Each of these phases will be explained, and the experimental evaluation approach for each phase will also be highlighted. In the Chapter Four, the construction algorithm model is explained together with the experiments' results and analysis. Chapter Five discusses the findings and outcomes of the improved HSA method, including a description of the analysis and discussion of the enhanced HSA approach. The hybridization of the improved HSA with GDA is demonstrated in Chapter Six along with the description of the algorithm for GDA modification, a modified method for the enhanced of HSA with GDA, experiments, findings, and a discussion. Chapter Seven describes the details on producing the real-world school timetable problem datasets based on the XHSTT format. In addition, the experimentation and results of solving the real-world HSTP dataset will be described using the proposed constructions and improvement methods. Finally, Chapter Eight outlines the overview of how each research objective was tackled, detailing its contributions, achievements, limitations, and prospects for future research.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

This chapter provides the reviews related to this research as illustrated in Figure 2.1. Firstly, the types of educational timetabling problems are described in Section 2.2. Next, section 2.3 explains in detail about HSTP which consists of problem definition, constraints and datasets. The methodologies for solving HSTP that have been used since 2010 are highlighted, together with the best results of those methodologies including the construction approaches. As the investigations of the construction approach will involve the heuristic rules, the Markov Chain is reviewed in section 2.5 to obtain information on decision making process. Sections 2.6 reviews the previous works that applied the metaheuristic of HSA, GDA, and its hybridizations. Section 2.7 reaches a uniform proposal with discussion. Section 2.8, literature gap and justification are demonstrated. Finally, section 2.9 summarizes this chapter.

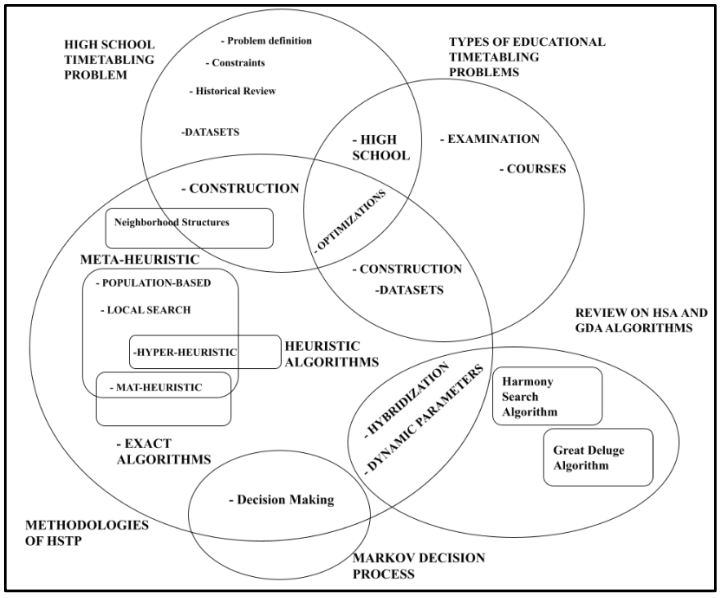


Figure 2.1. The theme of chapter two

2.2 Educational Timetabling Problems

The educational timetabling is categorized as twofold big class of problems which consists of High School Timetabling Problem (HSTP) and University Timetabling Problem (UTP). The UTP has two subclasses which are Course Timetabling Problem (CTP) and Examination Timetabling Problem (ETP) (Alencar et al., 2020). In general, the problem of timetabling is acknowledged as a nondeterministic polynomial-time complete (NP-complete) problem, which makes it complicated to offer a general optimal solution for a broad range of instances.

The aims of ETP is to avoid students for taking more than one examination slot at the same time and the examination slots should evenly spread during exam period (Valouxis et al., 2012). The well-known datasets of ETP came from University of Melbourne, University of Nottingham and University of Toronto that called as Carter datasets which has 12 instances (Gashgari et al., 2021; Qu et al., 2009). Methodologies have been used to solve ETP includes Graph Colouring, Genetic Algorithm, Tabu Search, Mixed Integer Linear Programming, Fuzzy Integer Linear, Fuzzy Multiple Heuristic Orderings, and Simulated Annealing (Gashgari et al., 2018, 2021).

The CTP allocates lecturers and lessons to timeslots, rooms, or additional facilities, matched to an individual student (Gashgari et al., 2021) which aim to reduce lectures and students overlaps (Valouxis et al., 2012). The well-known datasets of CTP is ITC-2007 that contains 24 instances (Babaei et al., 2015). Approaches that have been used to solve CTP involves Metaheuristic, Operational Research (OR), Multi-criteria, Intelligent Novel, and Distributed Multi Agent methods (Babaei et al., 2015).

HSTP includes the availability of teachers and classrooms, the amount of lessons that must be provided by teachers to particular classes or students, and a list of constraints. The well-known dataset of HSTP is XHSTT which consists of 39 data instances that were contributed by 13 countries such as Australia, Brazil, Denmark, Finland, Spain, Greece, Italy, Kosovo, Netherlands, United Kingdom, United States of America and South Africa. Approaches that have been used to solve HSTP involves Mathematical Optimisation, Graph Colouring, Heuristics, Matheuristics, Hyper-Heuristics, Hybrid Approaches and Metaheuristics algorithms (Tan, Goh, Kendall, et al., 2021).

Table 2.1 summarizes two key differences between HSTP and UTP. HSTP and UTP differ in parameters, constraints, and their impact on educational policies and institutional structures. The HSTP involves more parameters, such as resources and events, which are further divided into sub-parameters like students, teachers, and classes or rooms for resources, and meets, times, or slots for events. In contrast, the UTP focuses on parameters like courses/exams, rooms, and students, with fewer sub-parameters such as time slots for courses/exams in the course and exam timetabling problems. Constraints associated with each problem type also differ significantly. The HSTP typically deals with local policies and institute structures, emphasizing the need to allocate resources effectively within a single school. This involves ensuring no student or teacher is scheduled to be in two places simultaneously and that classrooms are available. On the other hand, the UTP addresses variations in the number of students across different institutions, which can significantly impact scheduling complexities. While the policies and institutional structures for UTP might be similar across universities, the varying number of students introduces additional challenges. The differences in parameters and constraints between HSTP and UTP reflect their

respective impacts on educational policies and institutional structures. HSTP focuses on optimizing schedules within a single high school, adhering to local policies and constraints. In contrast, UTP involves managing a more complex array of resources and constraints across multiple departments and campuses, aiming to accommodate diverse course requirements and student numbers. These distinctions highlight the varying levels of complexity and challenges in scheduling for high schools versus universities. Thus, this study chooses the HSTP to be further investigated since this problem is challenging and complex than UTP. They require sophisticated optimization techniques and algorithms to create schedules that meet all constraints and preferences while maximizing efficiency and resource utilization (Bashab et al., 2020; Gashgari et al., 2021; Pillay, 2014).

Table 2.1

Facts of High School and University Timetabling Problems

Idiosyncratic Facts	**High School (HSTP)**	**University (UTP)**	
		Courses (CTP)	**Examination (ETP)**
Complexity of Parameters (Gashgari et al., 2021)	The process of allocating resources, such as students, teachers, and rooms, to a set of events (Tan et al., 2021)	Allocating a collection of courses to a limited number of time slots and specific rooms with particular features (Tan et al., 2021)	Allocating a group of exams to a specific set of time slots and rooms with limited capacity (Tan et al., 2021)
Educational policies and institute structures (Gashgari et al., 2021; Pillay, 2014)	The majority of the constraints for these problems are influenced by local educational policies and institutional structures. It is worth noting that each set of problem constraints is unique and varies from one problem to another (Gashgari et al., 2021)	The majority of the constraints for these problems are influenced by international educational policies and institutional structures. This is due to the variation in the number of students across different institutions, which is a significant factor in these problems (Bashab et al., 2020)	

It is found that the HSTP has less number of publication compared to UTP where UTP has more 85% of volume over total publications related to the educational timetabling problem (Gashgari et al., 2021). Thus, due to these factors, the HSTP is worth to be investigated further. The subsequent section discusses on HSTP in more details.

2.3 High School Timetabling Problem

HSTP requires a certain of meet for a set of resources and time for specific students with a set of constraints. In assessing the solution to this problem, the hard and soft constraint violations will be calculated. The hard constraints must be fulfilled in order to generate a feasible solution (Tan et al., 2021). For instance, students are unable to sit for two classes or to have two teachers at the same time. Conversely, the soft constraints should be satisfied as considerably as possible so that a better-quality timetable can be produced. Soft constraints usually must be not violated to a certain degree in a real-world situation (Tan et al., 2021). For instance, some teachers are preferred for assignment to some classes only, and some time slots (morning or afternoon sessions) are preferred for assignment to some classes only.

There are several terms used in HSTP such as class, resource, room, time group, timeslot, tuple, event and idle. Table 2.2 gives a brief explanation of these terms. A class represents one set of students which join a specific subject at the same time. A resource is usually a teacher, room, student, or a class that attends the events (Kingston, 2018; Tan et al., 2021). A room is the locations where events are occurred (Kingston, 2018). A time or timeslot is one indivisible (unable to split) period of time so-called duration that relate to run an event (Kingston, 2018). A time group is a set of times or

time slots which related to a day (Kingston, 2018; Tan et al., 2021). A tuple consists of a class, teacher and a room which obliges to a schedule in timetable period (Pillay, 2014). An event refers to a meeting between resources and could be divided into a number of sub-events or lessons (Kingston, 2018; Tan et al., 2021). An idle is a free period for a teacher (Pillay, 2014).

Table 2.2

Terminology in HSTP

Term	Description	Reference
Class	Represents one set of students which join a specific subject at the same time.	(Kingston, 2018; Tan et al., 2021)
Resource	Usually a teacher, room, student, or a class that attends the events.	(Kingston, 2018; Tan et al., 2021)
Room	Locations where events occur.	(Kingston, 2018)
Time/Timeslot	One indivisible period of time (duration) related to run an event.	(Kingston, 2018)
Time Group	A set of times or time slots related to a day.	(Kingston, 2018; Tan et al., 2021)
Tuple	Consists of a class, teacher, and a room which obliges to a schedule in timetable period.	(Pillay, 2014)
Event	Refers to a meeting between resources and could be divided into a number of sub-events or lessons.	(Kingston, 2018; Tan et al., 2021)
Idle	A free period for a teacher.	(Pillay, 2014)

The objective function is a mathematical expression used to indicate a goal in problems with optimization, either to be maximized or minimized (Russell et al., 2010). Constraints in these problems specify the acceptable solutions. These constraints may be "soft," denoting a preference rather than an absolute requirement, or "hard," requiring that they always be fulfilled (Bazaraa & Shetty, 1979). In a scheduling

scenario, for instance, a hard constraint would specify that no employee can work more than 40 hours per week, while a soft constraint might state that it is preferred that employees not work more than five days in one week (Russell et al., 2010).

The HSTP is evaluated by the objective function which the total of hard and soft constraints that violated in the timetable. The objective function for the HSTP is normally reported in two types of presentation (Pillay, 2014). The first type presents the sum of the hard and soft constraint violations as produced by Papoutsis et al. (2003) which was used by Tan et al. (2020) as shown in equation 2.1 below:

$$min \sum_{t=1}^{T} \sum_{j=1}^{COL_t} C_{i,j} X_{i,j} \quad (2.1)$$

Explanation:

- t: Represents a specific teacher.
- T: The total number of teachers in the system. The sum $\sum_{t=1}^{T}$ indicates that the objective function iterates over all teachers.
- j: Represents a specific timetable or timetable slot.
- COL_t: The number of different weeks or slots in the timetable for teacher ttt. The sum $\sum_{j=1}^{COL_t}$ indicates that the objective function iterates over all timetables for teacher t.
- $C_{i,j}$: This is the cost associated with timetable j for teacher t. This cost function typically quantifies the violations of constraints. Hard constraints are requirements that cannot be violated (such as teachers being unavailable at specific times), while soft constraints are preferences (such as preferring to teach at specific times).

- $X_{i,j}$: A binary variable that takes on the value:
    - **1** if timetable j is assigned to teacher t,
    - **0** otherwise.

Objective:

- The goal is to **minimize** the total sum of constraint violations (both hard and soft) across all teachers and timetables.

The objective function is to minimize the total of hard and soft constraint violations where: t: teacher, T: the number of teachers, j: timetable, COL_t: number of different week timetable, $C_{i,j}$: the cost of timetable j for teacher t and $X_{i,j}$: correlates with binary variable.

The second type presents the integer value of the hard and soft constraint violations. For example, a value of *1.00447* denotes that the total number of hard constraint violations is 1, and the total penalty value of soft constraint violations is 447. This type is used by majority of researchers in their results to compare with other researchers (Brito et al., 2012; Fonseca et al., 2012; Shambour et al., 2013, Fonseca et al., 2016; Fonseca et al., 2016b; Saviniec et al., 2013; Teixeira et al., 2019). The hard cost is always ranked ahead of the soft cost, meaning that solutions are ranked first on their hard cost and then on their soft cost (Kristiansen et al., 2015a). This type of objective function simply sums the penalty value for both hard and soft constraint violations. The equation 2.2 shows the calculation objective function for this type.

$$min \sum_{p \in c, d \in p} s_{p,d,c} \; \forall \; c \; \in C \tag{2.2}$$

Explanation:

- p: Represents the point-of-application, which could refer to the point in the timetable where a particular teacher or task is applied. Essentially, this is the entity (e.g., a teacher or a timetable slot) to which the constraints are applied.
- d: Represents deviation. This refers to how much the solution deviates from the desired state (e.g., a hard or soft constraint). For example, if a hard constraint requires that a teacher must be available at a certain time, any deviation from this availability would contribute to the total deviation.
- c: Represents the constraint that is being evaluated. Each constraint is a rule that needs to be respected, whether it's hard (cannot be violated) or soft (can be violated but should be minimized).
- C: Represents the set of constraints. This could include both hard and soft constraints that apply across the schedule or timetable.
- $s_{p,d,c}$: The penalty value associated with violating constraint c at point p with a deviation d. This value quantifies how much each violation of a constraint costs, either in terms of soft constraint penalties or hard constraint violations.

Objective:

- The goal of this function is to **minimize** the total penalty value $s_{p,d,c}$ across all points p, deviations d, and constraints c.
- The function sums up the penalties for each point of application p, across each deviation d from the desired schedule, for each constraint c.

The objective function is to minimize the total of hard or soft constraint violations where, p: point-of-application, d: deviation, c: constraint, $s_{p,d,c}$: penalty value of the

deviation and C: set of constraint. Based on this objective function for both of hard and soft constraint violations, this study's evaluation and analysis will be implemented.

2.3.1 HSTP Benchmark Datasets

There are several HSTP datasets that have been used and tested includes Greek, Brazilian, Lectio, OR-library, and XHSTT real-world data instances. The following subsections discuss on each of the benchmark dataset.

2.3.2.1 Greek high school

Greek high school (GHS) datasets are consisted of 11 real-world data instances that contain seven hard and three soft constraints. The uniqueness of this dataset was that it involved the co-teaching cases (Tassopoulos & Beligiannis, 2012b). The hard and soft constraints in GHS are as follows (Tassopoulos et al., 2020):

GHS Hard Constraints

H1. Each teacher, during any given period in a day must be available and can teach at most one class, while each unavailable teacher must not teach a class during a day.

H2. Each teacher must teach a class out of a specified number of classes for a predefined number of periods and only on days this teacher is available to the school.

H3. Each class, in any weekday, must not have a maximum number of teaching hours.

H4. Each class, for a provided period, must assign to one teacher at most, not including for the case of co-teaching.

H5. Each class must have a compact schedule during any provided day, while idle hours are authorized only at the end of each day.

H6. In some cases, two teachers ought to be assigned to the same period at the same class, for a certain number of periods. This is a kind of co-teaching.

H7. Specific teachers ought to concurrently teach some classes for a pre-defined number of periods. This kind of co-teaching is different from **H6** hard constraint.

GHS Soft Constraints

S1. Each teachers daily schedule is required to be condensed, with no free time allowed in between teaching sessions.

S2. The weekly schedules of all teachers must be filled equitably on the days that they are available to teach.

S3. No class should be taught the same topic more than once in a single day.

Table 2.3

Greek High School Datasets Properties

Instances	Lecturers	Lessons/ Classes	Teaching hours per week	Teachers involved in co-teaching	Co-teachings	Constraints
1	34	11	35	9	36	12698
2	35	11	35	17	67	14069
3	19	6	35	0	0	6395
4	19	7	35	6	31	7389
5	18	6	35	0	0	2062
7	35	13	35	17	70	14838
8	11	5	30	0	0	3941
9	15	6	{32, 34, 35}	0	0	5538
10	17	7	30	0	0	5983
11	21	9	{33, 34, 35}	0	0	9483

Source: (Tan et al., 2021; Tassopoulos et al., 2020)

The feature of GHS datasets consists of number of teachers, classes, teaching hours per week, teachers involved in co-teaching, co-teachings and constraints, as shown in Table

2.3. The GHS objective function $objF$ in Equation 2.3 aims to minimise the soft constraints.

$$objF = constraint_{s1} + constraint_{s2} + constraint_{s3} \quad (2.3)$$

2.3.2.2 Brazilian high school

Brazilian high school (BHS) datasets contain 41 real-world data instances (Saviniec & Constantino, 2017). There are eight constraints which consist of five hard and three soft constraints as follows:

BHS Hard Constraints

1. Each tuple obliges to assign exactly the number of weekly meetings.
2. Each tuple does not oblige to have more than maximum number of daily meetings.
3. A class obliges to attend exactly one meeting each time.
4. Teachers do not oblige to schedule to periods where they are unavailable.
5. A teacher obliges to teach one lesson each time.

BHS Soft Constraints

1. Each tuple ought to get at minimum twice classes a week.
2. The schedules of teachers ought to concentrate on a minimum set of days.
3. Idle periods during the schedule of teachers ought to be avoided.

The objective function of BHS datasets is based on Equation 2.4.

$$min \sum_{i=3}^{5} \alpha_i^{hc} . \beta_i^{hc} + \sum_{i=1}^{3} \alpha_i^{sc} . \beta_i^{sc} \quad (2.4)$$

The objective function is to minimize the total of hard or soft constraint violations where, $\alpha_i^{hc}\ and\ \alpha_i^{sc}$ are the penalty violations for hard and soft constraints respectively while $\beta_i^{sc} and\ \beta_i^{hc}$ are the number of times that the constraints (soft and hard) are violated (Saviniec & Constantino, 2017).

2.3.2.3 Lectio

Lectio datasets is a complex model of HSTP from 200 different high school in Denmark which has 100 real-life datasets (Sørensen, 2013). Where an event is not allocated to a timeslot or/and a room, a punishment ought to be required. The objective function of Lectio dataset is to minimize the total of constraints as shown in Equation 2.5.

$$\min \sum_{e,r,t} \alpha_{e,r,t} x_{e,r,t} \tag{2.5}$$

The Equation 2.5 denotes this punishment by $\alpha_{e,r,t} \in \mathbb{R}^+$ and the decision variable is $x_{e,r,t} \in \{0, 1\}$, which uses value (1) where an event *e* requires to place in timeslot *t* in room *r*, and (0) otherwise.

The Lectio constraints which are described by Sorensen & Stidsen (2013) are as follows:

1. Each event ought to be allocated one timeslot and one room.
2. Each entity is able to only join in single event in each timeslot.
3. Each room is able to only be allocated once to each timeslot.
4. An event may be inaccessible toward a particular timeslot or a particular room by the teachers or students.

5. Some events may require specific rooms, i.e., physical education or chemistry lectures.
6. A room cannot be assigned to an event unless a timeslot is also given to the event.

2.3.2.4 OR-library

OR-library datasets have been artificially generated (Pillay, 2014). The level of OR-library datasets problem is described as "hard" timetabling problems. There are five datasets as shown in Table 2.4. The feature of OR-library datasets consists of number of lecturers, locations, lessons or classes, times and tuples. There is no mathematical formulation represent the objective functions of OR-library datasets. The objective of the OR-library is to minimize the value of a cost function which evaluates a given timetable. This cost function measures the quality of the solution. An ideal or perfect timetable would have a cost of 0, indicating no clashes. The cost function evaluates the timetable based on the number of clashes, and these clashes can be categorized into three components: class cost, teacher cost, and room cost.

Table 2.4

OR-library datasets properties

Datasets	lecturers	locations	lessons or classes	times	tuples
hdtt4	4	4	4	30	120
hdtt5	5	5	5	30	130
hdtt6	6	6	6	30	180
hdtt7	7	7	7	30	210
hdtt8	8	8	8	30	240

Source: (Pillay, 2014; Tan et al., 2021)

The goal is to achieve the lowest possible value of this cost function, meaning the fewest number of clashes or conflicts in the timetable. The optimization process can be tailored by weighting the different types of costs, emphasizing some types of clashes as being more important than others (Abramson, 1991).

The constraints of OR-library as described in Pillay (2014) are as follows:

1) The whole class and teacher meetings should be arranged based on the necessary set of times.
2) It must be no class that conflicts.
3) It must be no teacher that conflicts.
4) It must be no room that conflicts.

2.3.2.5 XML High School Timetabling (XHSTT)

XHSTT dataset characterizes the HSTP (Kingston, 2022). The XHSTT is formatted by Extensible Markup Language (XML) schema which explains the HSTP real-world. The purpose of XHSTT is to contain the HSTP characters. The XHSTT instances have four XML tags:

1. <Times> - times are a group of periods in a day.
2. <Events> - events are meetings between times and resources which provide data, such as workload, duration, resource, and time. The timeslots that must be allocated to, the event or group of events have pre-assigned timeslots, is called duration (Fonseca et al., 2016). The set of events linked to the duration and resources is called workload. Events with a special characteristic are grouped together in *EventsGroup*.

3. <Resources> - resources involves of teachers, classes, rooms, and students. A group of classes is assigned for a set of particular teachers, such as mathematics or history.
4. <Constraints> - constraints define the requirements on the real-world problem. Sixteen constraints are discovered, as itemized in Table 2.5. Each constraint provides evaluation at one point of application (the determination of a single cost) producing a non-negative integer deviation. The cost is determined by using Equation 2.6.

$$Cost = Weight \times CostFunction(deviation) \quad (2.6)$$

The $Weight$ is defined by particular constraints based on real-world problem in a scale between 0 to 1000, and $CostFunction$ computes the deviation. There are three categories of $CostFunction(deviation)$: quadratic, step, or linear. Additionally, every constraint might classify as hard or soft depends on the data instances.

Table 2.5

Constraints in the XHSTT format

Constraints	Description
Assign Resource	sets that the solution events ought to be allocated with resources
Assign Time	sets that the solution events ought to allocate with times
Split Events	lists into a constrained number of sub-events
Distribute Split Events	identifies the limitation on the value of solution events of a particular duration in periods of the instance event
Prefer Resources	lists that several “resources” are more desirable for allocating to several solution resources
Prefer Times	lists that several “times” are more desirable for assignment to several solution events

Avoid Split Assignments	lists that the split assignments are not good for several event resources
Spread Events	specifies that the solution events of a set of events ought to be spread out in periods of time
Link Events	sets that several events ought to be allocated with the similar times
Order Events	sets that the times of double events ought to be in order
Avoid Clashes	lists that some resources ought to have no conflicts
Avoid Unavailable Times	specifies that certain resources are unattainable to join any events at several times
Limit Idle Times	lists the limits on the value of times wherever the resources may be idle
Cluster Busy Times	set the limits on the value of time groups during which a resource may be occupied
Limit Busy Times	sets the limits on the value of times through several set of times where a resource might be occupied
Limit Workload	identifies the limits on the overall workload (maximum and minimum) of solution resources

Source*:* (Kingston, 2022)

For example, Figure 2.2 shows the XML format of the HSTP that represent the real-high school dataset in Australia. Two main XML tags (lines 2 and 36750) present the *Instances* and *SolutionGroups* (from previous works). The instance has four XML tags (*Times* - lines 11, *Resources* - lines 150, *Events* - lines 1066, and *Constraints* - lines 6138). In addition, another one XML tag called *MetaData* - lines 4 provides the contributor information such as name, contributor, date, country, and description for Australia dataset.

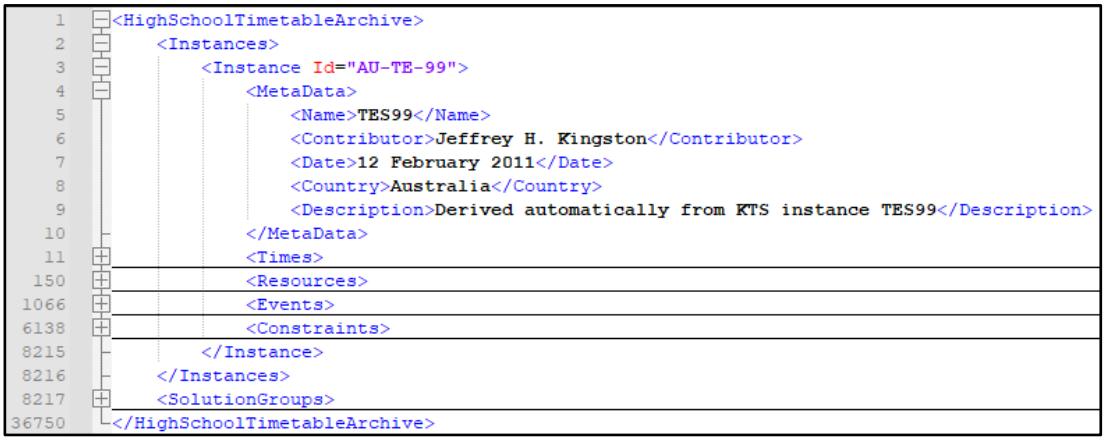


Figure 2.2. The XML format of high school in Australia dataset

Figure 2.3 shows the *Times* tag (lines 11) that has two XML tags which are *TimeGroup* and *Time*. The *TimeGroup* tag (lines 12) consists of days in a week which is defined by *Id* and *Name*, while the *Time* tag (lines 29) is defined by *Id*, *Name*, and *Day*.

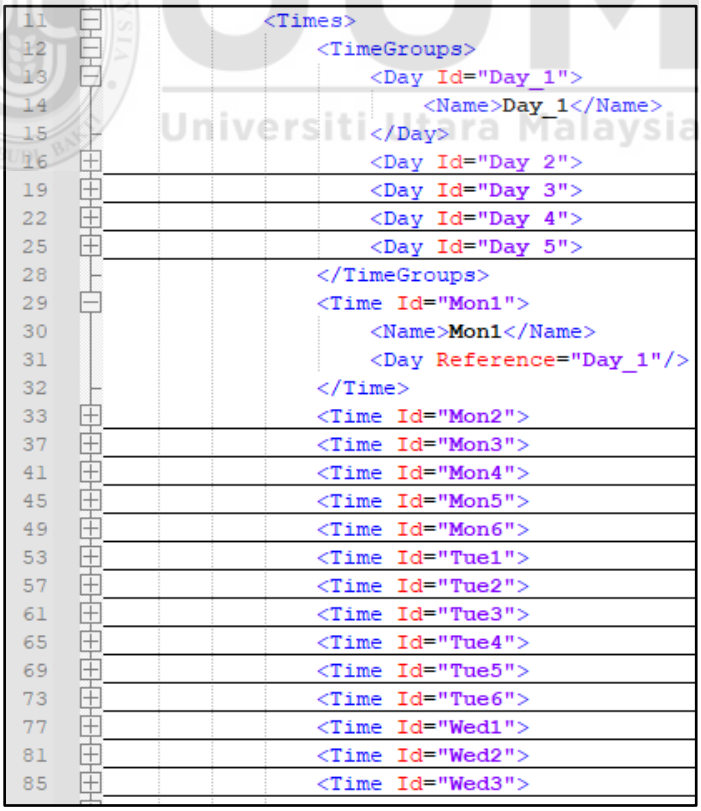


Figure 2.3. TimeGroups and *Time* examples in Australia dataset

Figure 2.4 illustrates the *Resources* tag (lines 150) which consist of *ResourceTypes*, *ResourceGroups* and *Resource* tags (lines 151, 162 and 316 respectively). The *ResourceTypes* tag contains the category of recourses which are Teacher, Room, and Class that is defined by *Id* and *Name*.

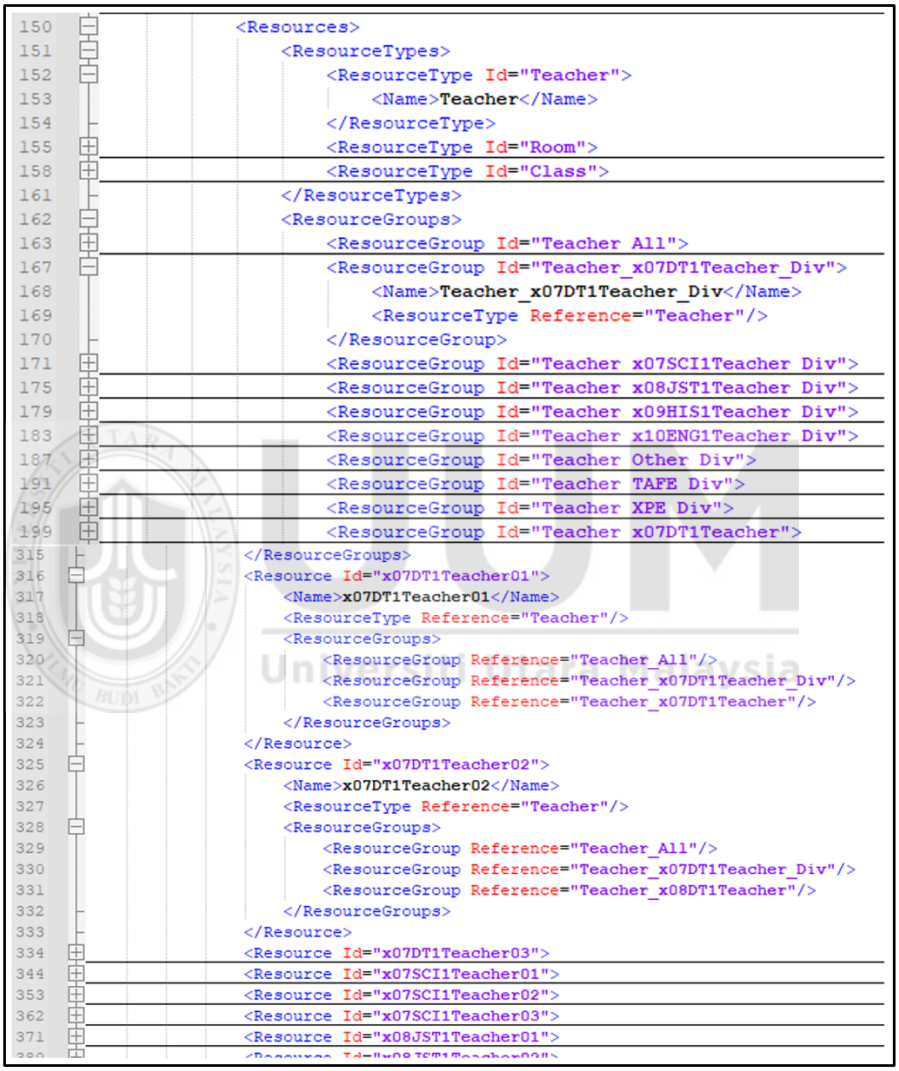


Figure 2.4. ResourceTypes, ResourceGroups and Resource examples in Australia dataset

As shown in Figure 2.5, the *Events* tag (lines 1066) which consist of *EventGroups* and *Event* tags (lines 1067 and 2146 respectively). The *EventGroups* tag contains *Course*

which is characterized by *Id* and *Name*. The *Event* tag is defined by *Id, Name, Duration, Workload, Course, Time, Resources* and *EventGroups*.

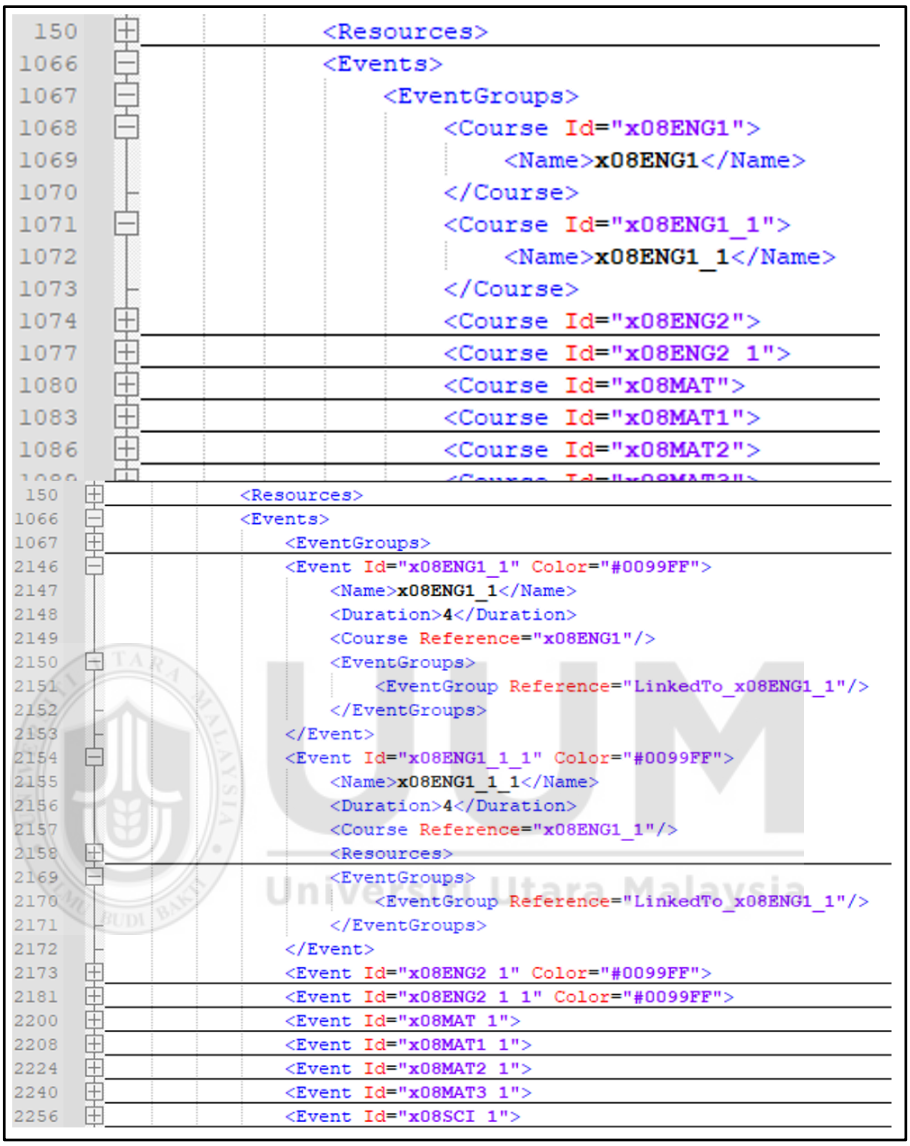


Figure 2.5. EventGroups, Course, Event examples in Australia dataset

XHSTT is contributed by 13 countries such as Australia, Brazil, Czech, Denmark, Finland, Spain, Greece, Italy, Kosovo, Netherlands, United Kingdom, United States of America and South Africa. These datasets are obtained from Benchmarking Project For

(High) School Timetabling[2]. As shown in Table 2.6, the name of the data instances of most countries consists of XX-YY-ZZ pattern in which XX represent the acronym of the country that contributed the data instances, YY represents the education institution and ZZ denotes the production year of the data instances.

Table 2.6

The XHSTT Date Instances

Country	Data Instances	Times	Resources	Events	Duration	Category
Australia	"AU-BG-98"	40	131	387	1564	M
	"AU-SA-96"	60	99	296	1876	M
	"AU-TE-99"	30	76	308	806	M
Brazil	"BR-SA-00"	25	20	63	150	S
	"BR-SM-00"	25	35	127	300	S
	"BR-SN-00"	25	44	140	350	S
	"BrazilInstance5"	25	44	119	325	S
	"BrazilInstance3"	25	24	69	200	S
	"BrazilInstance7"	25	53	205	500	S
	"BrazilInstance1"	25	24	21	75	S
Czech	"VillageSchool"	30	14	67	68	S
Denmark	"DK-FG-12"	50	438	1077	1077	L
	"DK-HG-12"	50	694	1235	1235	L
	"DK-VG-09"	60	262	918	918	L
Spain	"ES-SS-08"	35	91	225	439	S
Finland	"FI-PB-98"	35	111	387	854	M
	"FI-WP-06"	35	41	172	297	S
	"FI-MP-06"	35	64	280	306	M
	"SecondarySchool2"	40	79	469	566	M
	"ElementarySchool"	35	103	291	445	M
	"ArtificialSchool"	20	47	169	200	M
Greece	"GR-H1-97"	35	95	372	372	M
	"GR-P3-10"	35	114	178	340	S
	"GR-PA-08"	35	31	262	262	M
	"UniversityInstance3"	35	25	210	210	S
	"UniversityInstance5"	35	24	184	184	S
	"Preveza2008"	35	98	164	340	M
	"Aigio2010"	35	245	283	532	M
Italy	"IT-I4-96"	36	99	748	1101	M
Kosovo	"KS-PR-11"	62	164	809	1912	L
Netherlan-ds	"NL-KP-03"	38	587	1156	1203	L
	"NL-KP-05"	37	644	1235	1272	L
	"NL-KP-09"	38	194	1148	1274	L
	"Kottenpark2008"	40	126	1027	1095	L
	"GEPRO"	44	1102	2675	2675	L
UK	"UK-SP-06"	25	202	1227	1227	L
USA	"US-WS-09"	100	242	628	6354	M
	"ZL-LW-09"	148	37	185	838	S

[2] Benchmarking Project For (High) School Timetabling (https://www.utwente.nl/en/eemcs/dmmp/hstt/)

South Africa	"ZA-WD-09"	42	70	278	1253	M

They are 39 datasets which are categorized based on the size (small, medium, and large) determined by the number of times, resources, events and durations (Tan et al., 2021). The smallest dataset is from Czech called *VillageSchool* dataset with 68 total durations, and the dataset from Netherlands named *GEPRO* is the largest dataset with 2675 total durations. In conclusion, the XHSTT dataset was used in this work for thorough instance analysis; more details are given on this in the following section.

HSTP reveals distinct variations in constraints and complexity across datasets as shown in Table 2.7. The **Greek High School (GHS)** dataset, with seven hard and three soft constraints, introduces the complexity of co-teaching scenarios and aims to create compact schedules while minimizing soft constraint violations. Similarly, the **Brazilian High School (BHS)** dataset features five hard and three soft constraints, focusing on balancing teacher workloads and minimizing idle periods through a penalty-based system. The **Lectio** dataset emphasizes the correct allocation of events to timeslots and rooms, with complexity stemming from managing real-world data from 200 schools. The **OR-library** dataset, though artificially generated, focuses on minimizing clashes among teachers, classes, and rooms, while the **XHSTT** dataset stands out for its flexibility, with 16 constraints and a sophisticated penalty system to optimize schedules across various countries. Each dataset brings its own challenges, with XHSTT being the most adaptable and detailed, accommodating real-world scheduling complexities. The XHSTT is a comprehensive and flexible formulation used to address the complexities of high school scheduling. According to Tan et al. (2021), this dataset is structured in XML format and includes a variety of constraints—both hard and soft—

that allow for flexible modeling of real-world scenarios across different educational systems. It has been widely adopted for algorithm testing, particularly following its use in the International Timetabling Competition (ITC2011). Ceschia et al. (2023) further elaborate on XHSTT, noting that it is the most popular high school timetabling formulation due to its complexity and rich diversity of instances, drawing attention from researchers globally. This dataset includes detailed information on resources such as teachers, rooms, students, and events, making it a challenging and comprehensive benchmark. Both sources agree that XHSTT's ability to model a wide range of real-world scenarios makes it a valuable tool in educational timetabling research, although its complexity demands sophisticated solution methods.

Table 2.7

Summary of high school timetabling datasets, highlighting key constraints and complexity across various real-world and artificial instances

Dataset Type	Hard Constraints	Soft Constraints	Complexity Description
Greek High School (GHS)	7 hard constraints, e.g., each teacher can teach only one class per period, no exceeding the max number of periods.	3 soft constraints, e.g., teachers' schedules should be condensed without free time between teaching sessions.	Includes co-teaching cases and seeks to minimize soft constraint violations. The compactness of the schedule is a major focus.
Brazilian High School (BHS)	5 hard constraints, e.g., each teacher teaches only one lesson at a time and must avoid unavailable periods.	3 soft constraints, e.g., teachers' schedules should concentrate on fewer days, and idle periods should be minimized.	Minimizes hard and soft constraint violations using a penalty system. This dataset focuses on balancing teacher workloads.
Lectio (Denmark)	6 hard constraints, e.g., each event must have a timeslot and room, and no overlaps for rooms or events.	None mentioned	Focuses on punishment if events are not allocated to timeslots or rooms. A complex system with real-world datasets from 200 schools.
OR-library	4 hard constraints, e.g., no teacher, class, or room conflicts.	None explicitly mentioned	Artificial dataset with the aim to minimize clashes between teachers, classes, and rooms. High complexity is implied due to multiple categories of clashes.

XHSTT (XML format)	16 constraints including hard and soft, e.g., assigning resources, avoiding clashes, and limiting idle times.	Penalty functions applied for constraint violations (quadratic, step, or linear).	A highly flexible and detailed dataset using XML formatting, includes 13 countries with varied data and a comprehensive set of constraints.

2.4 Methodologies for Solving HSTP

In this section, an examination of the various methodologies that have been employed to address the HSTP is presented. Specifically, the sub-sections 2.4.1 and 2.4.2 provide an overview of the primary methods for solving HSTP, which include both construction and improvement approaches. Subsection 2.4.3 delves into the examination of the neighbourhood structures utilized in the improvement approaches, while subsection 2.4.4 presents the most best methodologies employed within the improvement approaches.

2.4.1 Construction Approaches

The construction stage is the step in producing feasible timetable (initial solution) that can be improvised later (Pillay, 2014). There are several methods used to produce the initial solution of HSTP such as GA, GRASP, THOR and KHE algorithm.

Filho & Lorena (2001) used GA to construct initial solution of HSTP. There were four processes such as a) represent the structure of HSTP such as pairs of teacher and class by binary columns. b) the GA was modelled as bi-objective optimization problems. c) conduct the evolution process in the GA, d) perform the selection, recombination and mutation to improve the solution. Two Brazilian high schools' datasets are used, and their results were considered successful. Another study (Raghavjee & Pillay, 2008) used GA with OR-library dataset to discover a solution for HSTP as constructive and improvement method. This method is alternative of *n* times to create *n* different

timetables of solution population. Their results were able to improve feasible timetables for all OR-library datasets. Wilke et al. (2002) used GA with German High School dataset which contains 1,309 students, 46 classes, 45 educational groups, 113 teachers, and 100 rooms for both constructive and improvement methods. Their population initial results led to several infeasible timetables that were able to be enhanced during the GA evolution procedure.

Other approaches for constructing initial solution for HSTP is GRASP. This approach constructs a timetable which is a step at a time using the notion of allocating first extremely important lessons in the extremely applicable times. The GRASP was used with the Brazilian high schools' datasets which contains several types of unfeasibility, and it was improved by Tabu Search heuristic with two different memory-based diversification strategies. Their performance greatly outperforms a prior hybrid tabu search method, and it has the benefit of a more straightforward design (Santos et al., 2005).

Another method to construct a solution is THOR which was created primarily to cater to Portuguese schools at various levels of education. There were three basic blocks (a graphical user interface, an automatic scheduler and a relational database) that make up its modular implementation. Two distinct and complimentary algorithms were included in the automatic scheduler such as a fast SA implementation-based iterative algorithm and a constructive algorithm using heuristics. THOR was used by 100 schools in Portugal with significant success (Melício et al., 2006).

Another approach for constructing the initial solution for HSTP is KHE that was produced based on several works of Kingston on HSTP structures from 2005 to 2014 (Kingston, 2005, 2007, 2012, 2014). The composition of KHE algorithm consists of structure, times assignment, resources assignment and clean-up stages. KHE is related to only XHSTT datasets. Fonseca et al, (2012) used KHE with late acceptance hill-climbing algorithm which used 18 XHSTT data instances. This approach obtained the best result in 14 out of 18 instances.

Table 2.8 lists comparison between CGA, GRASP, THOR and KHE. There are several practical and theoretical factors distinguish KHE from other methods. For practical aspects, firstly, it can handle cross-world datasets using XHSTT datasets. Secondly, KHE and XHSTT were agreed upon by the researcher's community as the state-of-the-art methods for optimizing HSTP in 2012 and 2014. Thirdly, KHE is comparatively easier to work with and many recent works on HSTP in the researcher's community use KHE as the construction initial solution for their optimization techniques. Fourthly, online documentation exists that explains how KHE algorithms work. For theoretical aspects, KHE provides better efficiency and can handle various dataset sizes, making it an appropriate choice for complex HSTP problems. Moreover, the other methods tend to offer poor efficiency and may reduce the likelihood of discovering an optimal or nearly optimal solution. Additionally, these techniques may require massive searching especially when dealing with difficult problems or large problem size. Therefore, based on the practical and theoretical features, KHE appears to be a more appropriate and efficient option for constructing initial solutions in HSTP compared to other available methods. Thus, this study used KHE as a construction algorithm for HSTP based on the above-mentioned supporting factors.

KHE algorithms consist of four phases that are depicted in Figure 2.6. However, the challenge lies in phase two which the algorithm needs to sort the layers before assigns the time to the layers.

Table 2.8

Comparison Facts of Construction Algorithm for HSTP

	Facts	CGA (Filho & Lorena, 2001)	GRASP (Santos et al., 2005)	THOR (Melício et al., 2006)	KHE (Kingston, 2014)
Practical	Cross-world datasets (Post et al., 2012; Tan et al., 2021)	No	No	No	Yes
	Used by the researcher's community in 2012 (Post et al., 2012)	No	No	No	Yes
	Used by the researcher's community in 2014 until now (Post et al., 2014)	No	No	No	Yes
	Easy to work with state-of-the-art methods based on previously existing methodologies (Tan et al., 2021)	No	No	No	Yes
	Documentation available	No	No	No	Yes
Theoretical	Good quality of solution	No	No	No	Yes
	Require extensive searching	Yes	Yes	Yes	No
	Capable of handling datasets of different sizes	No	No	No	Yes

To provide a critical discussion of the constructive approaches used for HSTP, it is important to explore each approach, focusing on their results, pros, and cons. Below is an analysis for the CGA, GRASP, THOR, and KHE methods. Pros and Cons of these constructive methods is shown in Table 2.9. CGA provides versatility and robust solutions but can suffer from infeasible initial outputs and a high computational cost. GRASP offers a simple yet efficient approach for smaller datasets, but its local optimization focus may require hybrid methods for better global performance. THOR

is user-friendly and integrates multiple algorithms for iterative improvement, but its complexity may require expert knowledge, and it is less efficient with larger datasets. KHE stands out for its efficiency and strong community backing, but its narrow focus on specific datasets (XHSTT) and challenges in layer sorting can limit its broader applicability. By analyzing the strengths and weaknesses of each method, KHE emerges as a highly efficient solution for certain datasets, but other methods may still be preferable for different contexts, especially when flexibility and scalability are required.

Table 2.9

Pros and Cons of Various constructive methods

Method	Pros	Cons
CGA (Filho & Lorena, 2001)	- Adaptability: Applied to various datasets of different complexities (Brazilian, OR-library, German). - Bi-objective Optimization: Handles multi-objective problems (teacher preferences and timetable conflicts). - Improvement Potential: Can iteratively improve solutions.	- Infeasible Initial Solutions: GA may produce infeasible timetables initially (Wilke et al., 2002). - Computational Cost: Requires many generations and large population sizes to converge, especially for large datasets. - Randomness in Results: Stochastic nature may lead to inconsistent results without fine-tuning.
GRASP (Santos et al., 2005)	- Efficiency: Allocates critical lessons first, handling essential elements optimally. - Straightforward Design: Simple and easy to implement, especially with Tabu Search. - Improvement with Hybrid Methods: Performance improves when integrated with memory-based strategies like Tabu Search.	- Initial Greed: Focuses on local optima, potentially missing better global solutions. - Requires Further Enhancements: Needs hybrid methods for large or complex datasets. - Narrow Focus: May result in suboptimal allocations for less critical tasks.
THOR (Melício et al., 2006)	- Modular Implementation: Features a graphical interface, automatic scheduler, and relational database.	- Complex Setup: Requires specialized knowledge for effective operation.

	- Iterative and Constructive Algorithms: Combines fast initial solutions with iterative improvements (simulated annealing). - Wide Usage: Successfully used in many schools across Portugal, proving its viability.	- Initial Feasibility Issues: Like GA, may generate infeasible timetables initially. - Less Efficient with Larger Datasets: May struggle with very large datasets beyond school timetabling.
KHE (Kingston, 2014, 2024)	- State-of-the-Art Performance: Recognized as a leading solution for HSTP, handling complex and cross-world datasets. - Wide Community Support: Well accepted by researchers and highly accessible. - Documentation: Extensive online documentation makes it approachable. - Efficiency with Large Datasets: Handles varying sizes, including large instances (e.g., Greece, Italy, Netherlands).	- Handling Complexity: May not work as effectively on datasets significantly different from XHSTT. - Dependency on Dataset Type: Limited generalizability to other dataset types or timetabling problems. - Higher Complexity in Layer Sorting: Phase two (layer sorting and time assignment) presents challenges, especially with diverse constraints.

To address the sorting issue in KHE algorithms, a heuristic-based approach is used to sort the layers. The sorting method prioritizes duration as the primary factor for sorting layers, without considering other layer attributes such as demand and index. This approach is employed because the most crucial factor is ensuring that the meets of one layer do not overlap in time, which is achieved by maintaining a minimum-cost matching of meets to times, as shown in Table 2.10 (Kingston, 2014).

The minimum-cost matching approach to meet assignment assumes that the cost of each meet assignment in a particular layer is independent of the other layers. This approach is crucial in obtaining a good solution, as noted by Kingston (2014). For instance, suppose there are two layers, A and B, with the following attributes: A (duration: 10, demand: 5, and index: 10) and B (duration: 9, demand: 15, and index: 16). In this case,

the sorting method based on heuristic rules will prioritize layer A for time assignment based on duration alone, without considering other attributes like demand and index. However, layer B is more complex than layer A because it requires the assignment of higher demand times to the meetings. Therefore, it is important to consider all relevant attributes when deciding which layer to be first assigned the time in order to achieve a minimum-cost matching of meets to times, rather than focusing solely on one aspect such as duration.

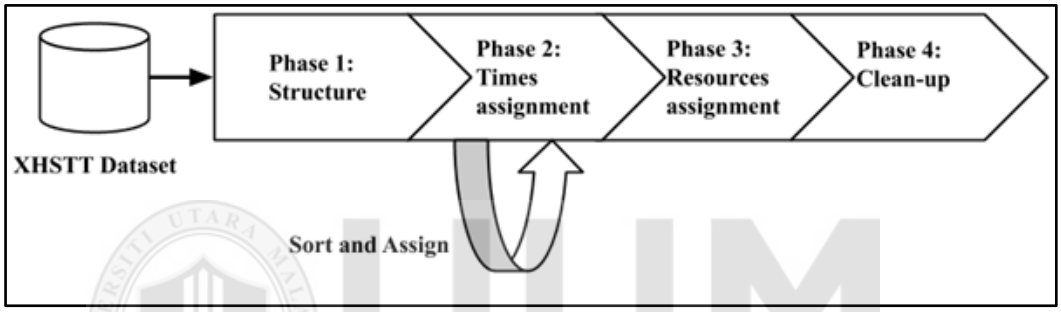


Figure 2.6. Phases of KHE construction algorithm

Table 2.10

Sorting Priority of KHE Algorithm's Time Assignment

Facts	Priority	Description	Linked to main fact
Duration	1	The number of layer slots (Kingston, 2021; Kingston, 2014)	The fact that the meets of one layer should not overlap in time by maintaining a minimum-cost matching of meets to times (Kingston, 2014)
Demand	2	The number of meetings and demand times (Kingston, 2021; Kingston, 2014)	
Index	3	The settings are based on constraints in the structure phase (Kingston, 2021; Kingston, 2014)	

2.4.2 Improvement Approaches

This section describes the approaches for solving HSTP based on mathematical optimisation, metaheuristic, matheuristics and hyper-heuristic.

2.4.2.1 Mathematical Optimisation Algorithms

Mathematical optimisation algorithms limit some or all of the problem variables to integer values and make assumption that the objective function and the constraints are linear (Tan et al., 2021). Examples of this algorithm includes integer, mixed integer and constraint programming.

Integer Linear Programming (ILP) or Integer programming (IP) is a mathematical technique that can be defined as follows: (a) a linear equation expressing the relationship between an appropriate quantity such as cost and the system's basic parameters, and (b) a set of linear equations or inequalities expressing the system's constraints or variable interactions (McNamara, 1971). Column generation (CG) approach which uses IP relaxation of the original IP problem applied to represent timetable for Greek high schools, and very high quality of feasible solution results are obtained (Papoutsis et al., 2003). Two approaches of IP were applied to resolve the problem of class blocking in which the Australia dataset is used. Their approach was able to solve the problem in a matter of seconds (Boland et al., 2008). Birbas et al. (2009) used IP approach to construct and optimize Greece dataset of HSTP by using problem formulation. Their results are conflict free as well as highly compact for the teachers. A cut and CG algorithms used to solve Brazilian datasets of HSTP in which the lower bounds of three datasets were obtained (Santos et al., 2012). Sørensen & Dahms (2014) applied an IP in two separated smaller models that helps to reduce the number of variables significant. Their computational results demonstrate that decomposition approach for Denmark real-world data instances, is way more effective. A research proposed binary IP models for Vietnamese data instances to decrease the number of used variables and was able to be solved by available IP solvers (Ngoc et al.,

2014). Their numerical experiments demonstrated the impressive efficiency of their IP model. Fonseca (2017) proposed new cuts and reformulation for the current IP model for XHSTT data instances that provides four new best-known lower bounds.

Constraint Programming (CP): to solve a given problem with CP, it was first formulated as a constraint satisfaction problem in which some variables ranging over specific domains and constraints were introduced. Next, the constraints were expressed in a small subset of first-order logic model (Apt, 2003). CP approach was firstly applied for solving HSTP by Yoshikawa et al. (1994) which constraint relaxation problem solver (COASTOOL) was used for solving three Japanese school datasets. Their findings show that COASTOOL is effective as a foundation for real-world applications in problems related to high school timetabling. Marte (2002) created and developed a constraint-based solvers method for constructing and optimize the 10 German HSTP dataset using the track parallelization problem. Without comparing their findings to alternative approaches, their findings on 1000 scheduling problems were statistically valid and practically applicable. Demirović & Stuckey (2018) formulated a novel CP model for HSTP via a scheduling-based point of view. Their experiments on 20 XHSTT data instances proved that CP approach beats other methods.

Mixed Integer Linear Programming (MILP) or Mixed Integer Programming (MIP): MIP approach is similar with IP approach, just that all variables are not integers (Huang et al., 2009). Kristiansen et al. (2015b) proposed a MIP model which solved in twofold stages with a business general-purpose. Their results demonstrated that MIP approach could obtain 2 optimum solutions for two datasets of XHSTT that previously not yet optimal. In addition, they also obtain optimal solution for 4 other XHSTT

datasets. Moreira (2015) described an approach based on MIP that helps the convergence of the searching process for solving the problem for some Brazilian schools XHSTT data instances. Tassopoulos et al. (2020) deals with HSTP for the case of 10 Greek school data instances which modelled as MIP and passed to Gurobi[3] and CPLEX[4] tools to be solved. Their approaches show great quality with one optimal solution. Saviniec et al. (2020) investigated a parallel metaheuristic-based algorithm and MIP formulations for solving 34 XHSTP data instances. Their MIP formulations are competitive with the best current MIP formulations for HSTP. In addition, their parallel algorithm offers excellent performance with alternative methods.

2.4.2.2 Metaheuristic Algorithms

Metaheuristic algorithm was defined as:

> *"A higher-level procedure or heuristic designed to find, generate, or select a lower-level procedure or heuristic (partial search algorithm) that may provide a sufficiently good solution to an optimization problem"* (Du & Swamy, 2016).

Metaheuristic could be categorised as local and population search approaches. The subsequent subsections will describe both approaches.

2.4.2.2.1 Local Search Algorithms

Local search algorithms preserve a single solution to adjust the solution with using specific neighbourhood-moves to investigate the zone around the present solution (Tan et al., 2021). Local search algorithms include simulated annealing, variable

[3] Gurobi Optimization https://www.gurobi.com/
[4] IBM CPLEX Optimizer https://www.ibm.com/products?q=CPLEX

neighbourhood search, very large neighbourhood search, iterated local search, tabu search, parallel local search and great deluge algorithms. This section describes several previous works that applied local search algorithms for solving the HSTP using different datasets.

Simulated annealing (SA): SA is one of the enormous usage algorithms to solve HSTP. The SA is a probabilistic single-solution-based metaheuristic search method (Du & Swamy, 2016). The SA with newly designed neighbourhood structure for HSTP that use a chain of exchanges between groups of timeslots as a replacement for exchanging two assignments was proposed (D. Zhang et al., 2010). The proposed SA was tested by using two Greek OR-library datasets of HSTP that performs better than existing approaches. Fonseca et al. (2012) proposed SA based approach with five neighbourhood structures using XHSTT dataset. Their approach able to achieve better results for five out of twenty datasets in a lower processing time. Brito et al. (2012) compared the SA with enormous types of VNS, such as Reduced Variable Neighbourhood Search (RVNS), and general variable neighbourhood search (GVNS) that use six neighbourhood structures tested on eighteen XHSTT data instances. According to their results, the variant of VNS methods generated good results compared to SA approach. Fonseca et al. (2016a) hybrid the SA with the Iterated Local Search (ILS) using twenty XHSTT data instances. Their hybridization local search approach (SA-ILS) was able to obtain feasible solutions for nearly all datasets. The SA was combined with Late Acceptance Hill-Climbing (SA-LAHC) with added mechanism named as stagnation-free (SA-sf-LAHC) (Fonseca et al., 2016). In the approach, it was found that the SA-sf-LAHC achieved the best findings compared to classical SA, SA-LAHC, and SA-ILS using eighteen XHSTT data instances. Teixeira et al. (2019) re-run

the SA and ILS from Fonseca et al. (2016a) for HSTP using fifteen XHSTT data instance with different experimentation settings. They managed to produce better results on two out fifteen datasets compared to original work of Fonseca et al. (2016a). Hao et al. (2021) proposed two phases of SA (minimize the hard and soft constraints respectively) for ten Chinese high school timetabling problems (CHSTPs) model. Their experimental results show the two-phase SA were applicable in solving the CHSTPs. The results on the above SA approaches for solving HSTP using XHSTT data sets is demonstrated in Table 2.11.

Table 2.11

Results Obtained Using SA Approach for XHSTT Datasets.

Algorithms	(Fonseca et al., 2012)	(Brito et al., 2012)	(Fonseca et al., 2016a)	(Fonseca et al., 2016)		(Teixeira et al., 2019)
	SA	SA	SA-ILS	SA-LAHC	SA-sf-LAHC	SA + ILS
Instances						
AU-BG-98	11.00375	11.00475	**4.00367**	-	-	6.00450
AU-SA-96	11.00011	18.00072	**10.0012**	-	-	14.0050
AU-TE-99	6.00148	9.00187	**5.00148**	-	-	7.00161
BR-SA-00	-	-	-	0.00102	**0.00078**	-
BR-SM-00	12.00150	12.00222	0.00103	2.04165	2.04164	**0.00091**
BR-SN-00	5.00258	4.00327	0.00156	0.00252	0.00221	**0.00149**
BrazilInstance5	6.00149	4.00295	0.00198	-	-	**0.00164**
BrazilInstance3	-	-	-	0.00174	**0.00160**	0.00264
BrazilInstance7	13.00285	11.00489	0.00294	-	-	**0.00012**
BrazilInstance1	-	**0.00050**	-	-	-	-
VillageSchool	-	-	-	-	-	-
DK-FG-12	-	-	-	-	-	-
DK-HG-12	-	-	-	-	-	-
DK-VG-09	-	-	-	-	-	-
ES-SS-08	-	-	-	0.00998	**0.00856**	-
FI-PB-98	4.00109	1.00838	**0.00000**	-	-	-
FI-WP-06	0.00074	0.00189	**0.00001**	-	-	**0.00001**
FI-MP-06	3.00114	0.00400	0.00102	-	-	**0.00086**
SecondarySchool2	-	-	-	**0.00000**	**0.00000**	-
ElementarySchool	-	-	-	**0.00003**	**0.00003**	-

Algorithms	(Fonseca et al., 2012)	(Brito et al., 2012)	(Fonseca et al., 2016a)	(Fonseca et al., 2016)		(Teixeira et al., 2019)
	SA	SA	SA-ILS	SA-LAHC	SA-sf-LAHC	SA + ILS
Instances						
ArtificialSchool	10.00008	19.00012	**0.00000**	-	-	-
GR-H1-97	**0.00000**	-	**0.00000**	-	-	-
GR-P3-10	0.00189	0.00149	**0.00000**	-	-	-
GR-PA-08	-	-	-	0.00007	**0.00006**	-
UniversityInstance3	-	-	-	**0.00005**	**0.00005**	-
UniversityInstance5	-	-	-	**0.00000**	**0.00000**	-
Preveza2008	**0.00000**	0.00198	**0.00000**	-	-	-
Aigio2010	-	-	-	0.00017	**0.00020**	-
IT-I4-96	-	-	-	0.00366	**0.00302**	-
KS-PR-11	-	-	-	4.65238	**6.36383**	-
NL-KP-03	0.05481	0.88387	**0.01189**	0.69608	0.489132	0.01409
NL-KP-05	0.02783	20.28482	**0.00963**	30.33310	30.233169	0.01078
NL-KP-09	-	-	-	**25.112948**	25.8112335	-
Kottenpark2008	-	-	-	**10.57476**	10.859939	-
GEPRO	2.06095	-	**1.00382**	-	-	1.00566
UK-SP-06	0.15208	2.49068	**0.11732**	-	-	0.18092
US-WS-09	-	-	-	-	-	-
ZL-LW-09	0.00020	0.00144	**0.00000**	-	-	0.00022
ZA-WD-09	-	-	-	**2.00012**	**2.00012**	-

Note: *Dash (-)* denotes that there are no results provided by the author.

In column six, hybridization of SA with modified LAHC via Stagnation-Free mechanism have good results on five data instances such as *BR-SA-00, ES-SS-08, GR-PA-08, UniversityInstance3,* and *IT-I4-96*, while *ElementarySchool* data instance have good solution results via SA-LAHC and SA-sf-LAHC. However, there are six data instances which are not yet solved either using individual or hybridization of SA such as *VillageSchool, DK-FG-12, DK-HG-12, DK-VG-09,* and *US-WS-09*.

Variable Neighbourhood Search (VNS): One of the most popular methods for solving the HSTP is VNS. The VNS combined neighbourhood structures and local search methods to handle search improvement. There are various forms of VNS that have been employed in earlier studies (Du & Swamy, 2016; Tan et al., 2021).

General variable neighbourhood search (GVNS) and reduced variable neighbourhood search (RVNS) were selected and organized according to neighbourhood structures computational complexities using seventeen or eighteen XHSTT data instances (Brito et al., 2012). The approach showed that the GVNS method produced great solutions for small data instances, while the RVNS method was effective for large data instances. Fonseca & Santos (2014) used basic VNS and VNS variants that hybridized with SA to solve HSTP using sixteen XHSTT data instances. The study found that both basic VNS and its variants algorithms produced good results compared to SA hybridization approaches. Fonseca et al. (2016b) used basic VNS algorithm for solving HSTP using twenty-four XHSTT data instances. It was found that the basic VNS algorithm produced good results compared to the initial solution. Teixeira et al. (2019) suggested three VNS algorithms for HSTP using fourteen XHSTT data instance that consists of skewed variable neighbourhood search (SVNS), skewed general variable

neighbourhood search (SGVNS), and adaptive VNS. The results showed that SGVNS performed equal with SA+ILS solver (Fonseca et al., 2016a) while superior to the GVNS and adaptive VNS algorithms.

Table 2.12 displays the outcomes of experiments done using VNS and its variant techniques on instances of XHSTT datasets. However, VNS approaches was never been applied to execute the *VillageSchool* dataset. In column seven, VNS by Fonseca et al. (2016b) outperforms the other VNS approaches on sixteen data instances, followed by SVNS (Fonseca & Santos, 2014) and SGVNS (Teixeira et al., 2019) (in columns six and nine respectively) that produce good results on seven data instances. GVNS (Brito et al., 2012) and RVNS (Fonseca & Santos, 2014), in columns two and five respectively, have not demonstrated any promising results when compared to the others.

Table 2.12

Results Obtained Using VNS Approaches for XHSTT Datasets

Algorithms	(Brito et al., 2012)		(Fonseca & Santos, 2014)			(Fonseca et al., 2016b)	(Teixeira et al., 2019)		
Instances	**GVNS**	**RVNS**	**VNS**	**RVNS**	**SVNS**	**VNS**	**GVNS**	**SGVNS**	**Adaptive VNS**
AU-BG-98	11.00475	11.00475	-	-	-	2.00398	4.00370	**1.00401**	7.00431
AU-SA-96	18.00052	18.00072	-	-	-	**4.00021**	12.00051	13.0046	17.0052
AU-TE-99	9.00187	9.00187	-	-	-	**1.00036**	7.00151	7.00163	7.00163
BR-SA-00	-	-	0.00040	2.00071	0.00035	**0.00022**	-	-	-
BR-SM-00	12.0123	12.00153	4.00108	21.00112	3.00132	3.00099	0.00094	**0.00090**	0.00094
BR-SN-00	4.00213	4.00213	0. 00157	6.00231	0.00145	**0.00104**	0.00148	0.00131	0.00163
BrazilInstance5	4.00148	4.00184	-	-	-	-	0.00158	0.00149	0.00165
BrazilInstance3	-	-	0.00113	2.00151	**0.00105**	-	-	-	-
BrazilInstance7	11.00267	11.00318	-	-	-	-	0.00249	**0.00248**	0.00282
BrazilInstance1	-	**0.00044**	-	-	-	-	-	-	-
VillageSchool	-	-	-	-	-	-	-	-	-
DK-FG-12	-	-	-	-	-	**0.01669**	-	-	-
DK-HG-12	-	-	-	-	-	**12.03371**	-	-	-
DK-VG-09	-	-	-	-	-	**2.02765**	-	-	-
ES-SS-08	-	-	0. 00907	0.03068	0.00783	**0.00415**	-	-	-
FI-PB-98	1.00049	1.00077	-	-	-	**0.00000**	-	-	-
FI-WP-06	0.00016	0.00073	-	-	-	0.00011	**0.00001**	**0.00001**	**0.00001**
FI-MP-06	0.00114	0.00129	-	-	-	**0.00084**	0.00087	0.00088	0.00087
SecondarySchool2	-	-	**0.00000**	0.00086	0.00003	-	-	-	-
ElementarySchool	-	-	**0.00003**	2.00007	0.00008	-	-	-	-
ArtificialSchool	19.00012	19.00012	-	-	-	**0.00000**	-	-	-
GR-H1-97	-	-	-	-	-	**0.00000**	-	-	-
GR-P3-10	0.00012	0.00020	-	-	-	**0.00003**	-	-	-
GR-PA-08	-	-	0.00006	0.00030	**0.00005**	-	-	-	-
UniversityInstance3	-	-	**0.00005**	0.00020	**0.00005**	-	-	-	-
UniversityInstance5	-	-	**0.00000**	2.00016	**0.00000**	-	-	-	-
Preveza2008	0.00037	**0.00033**	-	-	-	-	-	-	-
Aigio2010	-	-	0.00010	11.00200	**0.00008**	-	-	-	-
IT-I4-96	-	-	-	-	-	**0.00042**	-	-	-
KS-PR-11	-	-	-	-	-	**0.00004**	-	-	-
NL-KP-03	1.72413	0.85372	2.10217	2. 34766.0	2.10427	**0.01151**	0.01216	0.01281	0.01372
NL-KP-05	20.28710	20.28482	33.19059	35. 22914.0	33.18834	8.004460	0.00881	**0.00877**	0.01078
NL-KP-09	-	-	35.8543	45.148601.0	34.08649	8.008370	**0.00024**	**0.00024**	0.00042
Kottenpark2008	-	-	15.23962	36. 38936.6	**15.23756**	-	-	-	-
GEPRO	-	-	-	-	-	-	**1.00434**	1.00441	1.00532
UK-SP-06	2.48758	2.48450	-	-	-	53.01524	0.12542	**0.12466**	0.18418
US-WS-09	-	-	-	-	-	124.00000	-	-	-
ZL-LW-09	0.00078	**0.00074**	-	-	-	16.00000	-	-	-
ZA-WD-09	-	-	2.00008	10.00016	**2.00006**	3.00000	-	-	-

Note: *Dash (-)* denotes that there are no results provided by the author.

Large Neighbourhood Search (LNS) and Adaptive Large Neighbourhood Search (ALNS): In the large neighbourhood search (LNS) metaheuristic, a destroy and a repair technique implicitly defines the neighbourhood. While a repair technique reconstructs the destroyed solution, a destroy method destroys a portion of the current solution. Typically, the destroy method incorporates a stochastic component, which causes different portions of the solution to be destroyed each time the method is used. The ALNS heuristic extends the LNS heuristic by allowing several destroy and repair procedures to be utilized in the same search phase. The destroy/repair method is given a weight, which determines how frequently that approach is attempted during the search. As the search proceeds, the weights are constantly changed to ensure that the heuristic adjusts to the particular instance and the search's current condition (Pisinger & Ropke, 2019). Kristiansen & Stidsen (2012) applied ALNS to solve the student sectioning at Denmark high schools using their own seven different datasets. Two different types of moves: assign or unassign, eight different destroy methods, and four repair methods were proposed. Sørensen & Stidsen (2012) applied ALNS that used the destruct (ruin/remove) operator to the solution at hand, followed by the construct (recreate/insert) operator to repair the solution for six Denmark high school datasets. Sørensen et al. (2012) employed ALNS, which chooses and applies a remove-and-insert procedure to the candidate solution. Combining these techniques creates the categories of RemoveStrategy, AdaptiveStrategy, and AcceptStrategy, which describe the neighbourhood of the algorithm using all XHSTT data instances. However, they do not provided the results of their solutions. Demirović & Musliu (2017) combined local search with a novel maxSAT-based LNS which they used XHSTT data instance for their experiments. Their findings showed better results on 27 out of 39 XHSTT data

instances. As the works of LNS and ALNS that applied for solving XHSTT data instances were not much, the results comparison of solutions is not available.

Iterated Local Search (ILS): iteratively generates a series of solutions using a local search heuristic. An intermediate solution is produced after a new solution is generated via local search and changed by perturbation to escape from the local minima. A neighbourhood-based local search method is likewise intended to provide a better result. Determining which solution is chosen for continued evolution also includes a defined acceptance measure. If the new solution is of higher quality, it will take the place of the old solution. Until a termination requirement is satisfied, the process continues (Du & Swamy, 2016; Lourenço et al., 2010). Fonseca et al (2016a) applied hybrid metaheuristic based on SA and ILS for solving HSTP on twenty XHSTT data instances. Saviniec & Constantino (2017) proposed ILS method with two neighbourhood operators named as Matching (MT) and Torque (TQ). The ILS variants consist of ILS-TQ, ILS-MT-TQ and ILS-TQ-MT. The variant of ILS-MT-TQ produced good results compared to the other ILS variants on 11 out 34 data instances that collected from thirteen high schools in Brazil. In order to improve the ILS-TQ proposed by Saviniec & Constantino (2017) for solving HSTP, Jonathan (2019) changed the ILS-TQ based on SA cooling scheme inside or outside the local search for the 34 real-world datasets of Brazilian schools. However, their results did not improved well compared to Saviniec & Constantino (2017). Andrade et al. (2019) applied ILS that select a teacher whose schedule need some improvement. Then a subject is chosen on the day that is not preferred by the teacher. A new time, chosen at random within the teacher's choices, will be used for this subject. Two factors are examined for this change: the amount of time spent on this subject by the students and their teacher. These analyses

are done so that, after the process, a feasible solution can be achieved. This method was applied to solve the elementary and high school timetables from Brazil. The performance of ILS was better compared to the other two methods used on their works (mathematical model and local search). The results comparison of the solutions is not provided due to the small number of ILS works that were applied for solving XHSTT data instances.

Tabu Search (TS): TS is a method that is designed to solve optimization problems iteratively. It directly addresses use of memory structures and responsive exploration to guide a hill-descending heuristic so that exploration can continue without becoming confused by the lack of improvement moves (Glover & Laguna, 1997). Minh et al. (2010) proposed a Tabu Search-based algorithm that efficiently applied to 3 data instances of two high schools in Vietnam. In their approach, single moves are the most common type of move employed. Additionally, once a certain amount of unimproved moves have been made, swap moves and block-changing moves are used. The approach can be efficiently applied to the timetabling problem of two high schools. The obtained results outperform the handcrafted ones used in practice. Hooshmand et al. (2013) used TS with two types of neighbourhood searches: out-in and intra moves using three Iranian high school datasets. Out-in moves are always preferred, unless there have been no better solutions for a long time. In addition, a diversification technique is employed to enhance the method's exploration of previously unexplored areas of the search space and reduce the chance of becoming trapped in local optima. It is evident that utilizing diversification and intra moves both significantly reduce objective functions. The experiments have proven that this algorithm can create timetables that are nearly optimal and acceptable by the school management. Sanjay & Rajan (2017) applied

partial feasibility preserving GA (PFP-GA) with TS to offer a faster response time to the challenging timetabling problem using one HSTP hard timetabling (hdtt) dataset in OR-Library. By randomly switching room and teacher pairs inside a class, the TS generates the required number of neighbours from the initial solution. These pairs' indexes are kept in a list, or tabu list, whose size is flexible. When doing a swap operation to obtain the next neighbour, these indexes are avoided. If the new member's fitness is lower than that of the swapped neighbour, the process of swapping will result in the production of a different candidate solution. If the fitness score of the individual, or prospective candidate, with these movements is better than that of the best solution in the current iteration, then the movements from the tabu list were allowed. The entire operation is repeated a specified number of times. The optimal result of the tabu search is saved in an array for further use. In comparison to using just simple GA, the TS and PFP-GA combination produced results substantially faster. Due to the unavailable of TS works in solving HSTP using XHSTT data instances, the results comparison of the TS methods is not provided.

Parallel Local Search (PLS): PLS begins from a random configuration and aim to enhance its configuration by slight changes in the principles of the problem variables (Hamadi & Saïs, 2018). Saviniec et al. (2018) employed the parallelism strategies that contemplate two various parallel frameworks using 34 XHSTT data instances. The first is based on central memory, whereas the second is based on memories that have been diversified and intensified. They were able to consistently produce high-quality solutions for the given problem, and for two variants of it, their method surpassed state-of-the-art algorithms. The presentation of their results used the first type of objective function calculation which is the sum of the hard and soft constraint violations (as

described in section 2.3). For that reason, the results from this approach will not be included in the comparison. Furthermore, the comparison is not compatible from this study since they are using multiple processor and memories simultaneously.

Great deluge algorithm (GDA): GDA is a solution-based metaheuristic for continuous global optimization that accepts inferior suitable solutions when employing the solution search method that facilitates the escape from the local optima (Dueck, 1993). Mushi (2011) developed GDA based on an adaptation which utilizes a non-linear (NLGD) reduced rate in the decline of water level for solving three different high schools' datasets in Tanzania. Their method outperformed the basic GDA for all datasets. Kheiri & Keedwell (2017) utilized GDA, HC, SA, record-to-record travel (RR), and late acceptance (LA) methods as move acceptance components in their hyper-heuristics framework for solving HSTP using 26 XHSTT data instances. However, the performance of GDA with the selection hyper-heuristics was weaker compared to the other combination of move acceptance methods with selection hyper-heuristics. Due to the used of different datasets, these works will be not compared to each other.

2.4.4.2.2 Population-Based Algorithms

Population-based algorithms are a set of solutions which are iteratively updated till the termination requirement is satisfied (Du & Swamy, 2016). The following are the population-based algorithms that applied for solving HSTP such as evolutionary algorithms (EA), genetic algorithms (GA), swarm intelligence (SI) and harmony search algorithm (HSA).

Evolutionary Algorithms (EA): EA is a general class of optimization algorithms that use principles inspired by natural evolution, such as reproduction, mutation, and selection, to generate new solutions to a problem (Yu & Gen, 2010). Fernandes et al. (1999) presented a method for creating schedules for a high school using an EA. The schedules for classes, teachers, and classrooms are created to meet the needs of the school and comply with constraints for assigning lessons. The use of a problem-specific chromosome representation and a repair function during the evaluation of fitness allows the algorithm to stay close to valid solutions. Within this algorithm, "Bad Genes Mutation" operator was introduced. The technique was used to solve the DFL high school timetable, and it produced outcomes that were better or on par with those produced manually while also being completed significantly more quickly and with less effort. Hartmanis & Leeuwen (2001) outlined EA for creating school schedules, with a focus on the Italian school system. The authors improved upon heuristics to achieve strong generalization capabilities. Special attention was given to the problem formulation and the use of fuzzy logic, as well as testing various genetic operators and configurations. The algorithm was tested on real-world instances, resulting in successful and practical outcomes, which led to the creation of a commercial product. Domrös & Homberger (2012) proposed an EA for solving HSTP using XHSTT data instances. Their works managed to place themself as a finalist in the International Timetabling Competition 2011 (ITC2011). However, the solutions developed in round 1 do not correspond to the existing best known or optimal solutions. As no prior work has been done utilizing EA to address the HSTP using XHSTT data instances, it is not feasible to compare the performance of different EA methods.

Genetic Algorithms (GA): GA is a specific type of evolutionary algorithm that uses a genetic representation of solutions, such as strings of bits or real-valued numbers, and operations, such as crossover and mutation, that mimic the mechanics of biological inheritance (Deb, 1999). GA are the most well-known form of evolutionary algorithms (EA) (Du & Swamy, 2016). Raghavjee & Pillay (2010a) present GA via two-phase approach which are firstly to evolve objective function of timetable and secondly to enhance the solutions quality to solve HSTP using the South African data instances. They referred to their approach as an informed genetic algorithm (IGA) because it uses heuristics, a type of domain knowledge, to direct the search process. Raghavjee & Pillay (2010) in their other work, employed GA with tournament selection and mutation operators for solving HSTP using the South African data instances. This method produced a better-quality timetable compared to their results using IGA approach. Raghavjee & Pillay (2012) compared the implementation of GA and genetic programming (GP) for solving HSTP using Abramson (1991) data instances. GP demonstrated to be more successful than GA. Raghavjee & Pillay (2013) used GA to solve HSTP using the South African data instances. Their study discovered that several mixtures of low-level construction heuristics such as selection methods and genetic operators, generate feasible timetables of great feature for several HSTP dataset. Sutar & Bichkar (2016) used simple GA (SGA) and GA with probabilistic repair and knowledge augmented operators in the initialization, crossover, and mutation to expedite the search using hdtt4 of the OR-library data instances. The modified GA converged faster than SGA. As there is no existing research that has applied GA to solving the HSTP using XHSTT data instances, it is not possible to compare the effectiveness of various GA approaches.

Swarm Intelligence (SI): SI investigates the collective behaviours of autonomous, decentralized swarms that emerge through the local interactions of their individual components with one another and their surroundings (Du & Swamy, 2016). Swarm intelligence algorithms, in general, are designed based on interactions between living creatures such as ant colonies, bee colonies, bird flocking, hawks hunting, animal herding, bacterial growth, fish schooling, behaviour of cats and microbial intelligence. Related to these living organisms, the metaheuristics are ant colony optimization (ACO), artificial bee colony (ABC), particle swarm optimization (PSO), hawks optimization algorithm (HOA), herding optimization algorithm (HOA), bacterial foraging optimization (BFO), fish swarm algorithm (FSA), cat swarm optimization (CSO) and microbial genetic algorithm (MGA), respectively. In addition, particle swarm optimization (PSO) also relate to SI as its involved the behaviour of social animals, such as birds and fish (Angadi et al., 2021). However, only certain metaheuristics from this category were used to solve HSTP. Tassopoulos & Beligiannis (2012a) designed a new adaptive algorithm based on PSO for solving HSTP using Greek real-world data instances. The PSO algorithm was compared with GA, EA, column generation and constraint programming approaches. It was revealed that the PSO performed better in 66.7% of the cases and produced results that were equal in 22.2%. Furthermore, the PSO algorithm's close average and best outcomes across several runs provided further evidence of its stability and effectiveness. Katsaragakis et al. (2015) applied Artificial Fish Swarm (AFS) and PSO algorithms separately for solving HSTP using Greek datasets. They demonstrated these two SI algorithms performance and efficiency compared with EA, SA, Hybrid PSO and GA selection perturbative hyper-heuristic (GASPHH) algorithms. According on experimental findings, the proposed PSO algorithm produces the greatest outcomes ever discovered

for 5 over 6 instances of the Greek dataset. The proposed AFS algorithm, which represents the first attempt to apply an AFS algorithm to the school scheduling problem in the literature, nevertheless, also demonstrates highly satisfying results because it outperforms EA, SA and GASPHH approaches. Skoullis et al. (2017) used hybrid CSO based algorithm that incorporates a modified auxiliary tracing and seeking mode procedure for solving HSTP using Greek high schools data instances. The proposed algorithm was evaluated and compared against GA, EA, SA, PSO and AFS. The experimental results showed that the proposed algorithm outperformed other methods in eight out of ten instances, indicating its superior performance and potential for solving the HSTP. Additionally, the study found that the hybrid CSO based algorithm was user-friendly and used less computational time. In a recent study, Tan et al. (2021) proposed a novel algorithm for addressing HSTP using XHSTT data instances, which utilizes a combination of PSO and Hill Climbing (HC) techniques. This hybrid algorithm incorporates a unique solution transformation strategy and a particle elimination mechanism, which were shown to improve the performance of the algorithm. Empirical evaluations were conducted on the proposed algorithm and compared to the state-of-the-art methods. The results indicate that PSO and HC performs well in minimizing soft constraint violations but it is not as competitive in minimizing hard constraint violations. Given the lack of prior studies that have employed SI to address the HSTP using XHSTT data instances, it is not feasible to perform a comparative analysis of the performance of different SI methods in this context.

Harmony search algorithm (HSA): HSA was founded by Geem et al. (2001) and was motivated by jazz artists' use of improvisation. The only work that applied HSA for

solving HSTP is by Shambour et al. (2013) which hybridized HSA with SA for solving 12 XHSTT data instances. The performance comparison of the HSA-SA to the five competing approaches. The proposed approach was compared with SA, EA, hyper-heuristic search strategies, ALNS and VAGOS (benchmarking project). The approach proposed in the study was demonstrated to have the ability to generate solutions that are comparable to the highest quality solutions found in two instances, and solutions that are near to the highest quality solutions for the remaining instances.

2.4.2.3 Matheuristics Algorithms

Matheuristics is a field of study that combines mathematical optimization techniques with heuristic search methods to solve problems that are too complex for traditional mathematical optimization methods alone. This can be used in a variety of fields, including operations research, computer science, and engineering (Gomes et al., 2020). Dorneles et al. (2014) presented a new approach that uses a fix-and-optimize heuristic combined with a variable neighborhood descent (VND) method for solving HSTP using seven XHSTT data instances. The method is specifically designed to meet two requirements that are common in Brazilian schools: minimizing the number of working days and avoiding idle timeslots. The approach used different types of decompositions, such as class, teacher, and day, and it was able to find better solutions than general purpose solver known as CPLEX[5] for several instances. Fonseca et al. (2016b) presented a matheuristic approach that combines VNS algorithm with mathematical programming-based neighborhoods for solving HSTP using 17 XHSTT data instances. The results of computational experiments on benchmark instances show that this new

[5] IBM. ILOG CPLEX 12.1 User's manual. Mountain View, CA; 2009. URL http://www.ilog.com/products/cplex.

approach is more successful than the standalone VNS algorithm. Furthermore, the proposed algorithm improved 15 out of 17 current best-known solutions. Both works are not comparable as Dorneles et al. (2014) presented their results using the first type of objective function presentation which is the sum of the hard and soft constraint violations, while Fonseca et al. (2016b) used the second type of objective function presentation that used the integer value of the hard and soft constraint violations.

2.4.2.4 Heuristic and Hyper-Heuristic Algorithms

Heuristics algorithms belongs to experience-based techniques for learning and problem-solving. Particular, these techniques are problem-dependent and aimed to be only used for solution of a certain problem (Du & Swamy, 2016).

Moura & Scaraficci (2010) used two heuristics: a basic GRASP and a GRASP with path-relinking improvement for solving HSTP using three Brazilian school data instances. In their finding, GRASP with path-relinking was good in a computational time compared to basic GRASP. Veenstra & Vis (2016) applied three phase of approaches consist of simple rule-of-thumb, ILP and heuristic for solving HSTP using five different schools in Netherlands. The heuristic algorithm employed in this study begins by utilizing an initial schedule for all classes. This is followed by an iterative optimization of the schedule for each class, proceeding in a sequential manner from the first class to the last class. The optimization of each individual class schedule necessitates the implementation of three distinct steps. The first step involves the fixation of schedules for the remaining classes and the subsequent update of the availability matrix, such that the teacher's availability corresponds to the time slots designated for class in the initial schedule. The second step pertains to the actual

optimization of the schedule for the current class, which is achieved through the solution of an ILP problem. The final step includes the updating of the availability matrix, such that the teacher's unavailability is reflected in the time slots assigned to the class in the newly optimized schedule. This procedure is reiterated until each class schedule has been optimized exactly once. The heuristic outperforms the simple rule-of-thumb for data instances of school 2, 3, 4 and 5. Given that no prior research has utilized heuristics to address HSTP using XHSTT data instances, it is not feasible to make a comparison of the effectiveness of different heuristic strategies.

Hyper-heuristic is a strategy for solving problems that involves selecting or creating lower-level heuristics to solve a specific problem. It is a type of meta-heuristic approach that seeks to find good solutions by combining or modifying different heuristics, rather than relying on a single, fixed algorithm (Burke et al., 2013; W. Li et al., 2023). Pillay (2010) applied EA-based hyper-heuristic (EA-HH) for solving HSTP using five basic school timetabling problem. A comprehensive evaluation of various low-level construction heuristics was conducted, and EA was implemented to systematically search the space of heuristic combinations. The study also sheds light on the use of "non-destructive" mutation and crossover operators, which incorporate the utilization of hill-climbing, as a means to enhance the performance of the EA-HH. In addition, the performance of the proposed EA-HH was compared to other state-of-the-art methods such as GA, greedy search algorithm (GS), TS, SA and Hopfield neural networks (HNN). EA-HH was providing valuable insights into the effectiveness of the proposed approach. The EA-HH method may not have been designed to achieve the best results and may not be as effective as other optimization methods, however, its performance in solving the problem in question is similar to that of other methods. Braak (2012)

applied hyper-heuristic algorithm that combined grading, SA, reschedule resource, ejection chain and HC algorithms for solving HSTP which tested on 15 XHSTT data instances. In their approach, the hyper-heuristic is capable of determining which heuristic method should be used at any given time and also can adjust the parameters of each algorithm. This allows the strategies to adapt to the specific data they are working on, resulting in solutions of adequate quality. In Ahmed et al. (2015), the authors investigated the performance of 15 different selection hyper-heuristics. Each hyper-heuristic component has its own unique characteristics, some of which are adaptive methods, others are not, some are learning-based while others are not. The authors evaluate the performance of these hyper-heuristics by combining each selection method in Simple Random (SR), Random Permutation (RP), Random Descent (RD), Random Permutation Descent (RPD) and Choice Function (CF) with each acceptance method in improve or equal moves (IE), SA and GD on 18 XHSTT benchmark instances. The results show that the approach that incorporates an adaptive Random Permutation with GD move acceptance (RPGD) method is successful on the XHSTT benchmark instances, and this selection hyper-heuristic ranks second among previously proposed approaches. Kheiri et al. (2016) proposed a method called hyper-heuristic search strategies and timetabling (HySST) for HSTP using 21 data instances from XHSTT. HySST generated the best new solutions for three instances. They achieved this by using a standard stochastic (randomness) search method but significantly improved it by adding a selection hyper-heuristic with an adaptive acceptance mechanism that combines two hill climbing heuristics called intensification-stage. One of the operators is based on neighborhood structures using ejection chains, while the other is a type of first improvement hill climbing operator. The HySST method performed better than other methods such as EA, ALNS, and

VAGOS (benchmarking project), but did not surpass the GOAL approach method. A new sequence-based selection hyper-heuristic (SSHH) is introduced in Kheiri & Keedwell (2017) study, which used five different types of reusable metaheuristic-inspired methods as move acceptance methods, including HC, SA, GD, record-to-record travel (RR), and LA methods. The authors presented an easy-to-use, easy-to-maintain, and effective sequence-based selection hyper-heuristic that solves HSTP using 25 XHSTT data instances. Their results show that this method is able to find new best known solutions for a number of problems in the timetabling domain. Their study illustrated the advantages of sequence-based selection hyper-heuristics and demonstrates that they outperform existing state-of-the-art methods which are GOAL, HySST, ALNS and EA.

Table 2.13 shows the results comparison of solutions between hyper-heuristic methods that described above. The results Kheiri & Keedwell (2017) exhibit superior performance compared to the others except the results of BR-SM-00 and ZA-WD-09 data instances. However, for all methods of hyper-heuristic, there were five data instances that have not yet to be successfully solved including GEPRO, ArtificialSchool, VillageSchool, BrazilInstance1, and BrazilInstance5 data instances.

Table 2.13

Results Obtained Using Hyper-heuristic Approaches for XHSTT Datasets.

Algorithms / Instances	(Braak, 2012) Combination of Algorithms	(Ahmed et al., 2015) RPGD	(Kheiri et al., 2016) HySST	(Kheiri & Keedwell, 2017) SSHH
AU-BG-98	-	-	-	**0.00520**
AU-SA-96	-	-	-	**0.00002**
AU-TE-99	-	-	-	**0.00061**
BR-SA-00	-	0.00096	0.00044	**0.00010**
BR-SM-00	4.00165	10.00143	**0.00176**	2.00117
BR-SN-00	0.00169	0.00266	0.00150	**0.00101**
BrazilInstance5	-	-	-	-
BrazilInstance3	-	0.00152	**0.00084**	-
BrazilInstance7	**0.00274**	-	-	-
BrazilInstance1	-	-	-	-
VillageSchool	-	-	-	-
DK-FG-12	-	-	-	**0.01522**
DK-HG-12	-	-	-	**12.0263**
DK-VG-09	-	-	-	**2.02731**
ES-SS-08	-	0.01451	0.00920	**0.00517**
FI-PB-98	0.00081	-	-	**0.00008**
FI-WP-06	0.00042	-	-	**0.00007**
FI-MP-06	0.00120	-	-	**0.00089**
SecondarySchool2	-	0.00009	**0.00000**	-
ElementarySchool	-	0.00004	**0.00003**	-
ArtificialSchool	-	-	-	-
GR-H1-97	-	-	-	**0.00000**
GR-P3-10	0.00121	-	-	**0.00000**
GR-PA-08	0.00028	0.00019	0.00009	**0.00004**
UniversityInstance3	0.00013	0.00010	**0.00007**	-
UniversityInstance5	0.00011	0.00002	**0.00000**	-
Preveza2008	**0.00131**	-	-	-
Aigio2010	-	0.00596	**0.00218**	-
IT-I4-96	0.00176	0.00520	0.00052	**0.00038**
KS-PR-11	0.00264	17.09084	-	**0.00003**
NL-KP-03	-	0.40862	0.03919	**0.00466**
NL-KP-05	-	26.26129	15.28693	**0.00811**
NL-KP-09	-	22.99999	18.08010	**2.07495**
Kottenpark2008	-	24.99999	16.17720	-
GEPRO	-	-	-	-
UK-SP-06	-	-	-	**19.01294**
US-WS-09	-	-	-	**0.00512**
ZL-LW-09	1.0019	-	-	**0.00052**
ZA-WD-09	-	2.00279	**0.00013**	9.00000

Note: *Dash (-)* denotes that there are no results provided by the author.

In details, mathematical optimization algorithms like IP, CP, and MIP have been widely used in solving HSTP. IP works by formulating a linear relationship between costs and

system parameters while restricting variables to integer values. This approach has proven successful in solving datasets from countries such as Greece, Vietnam, and Australia, yielding optimal results. However, IP struggles with scalability and becomes computationally expensive for larger datasets. CP, on the other hand, is based on logic models and focuses on satisfying constraints. It has been applied in real-world datasets like Japan and Germany, though comparisons with other methods are sparse. CP offers flexibility but may slow down when handling complex datasets. MIP extends IP by allowing non-integer variables, making it more flexible and suitable for larger and more complex problems. It has demonstrated excellent results with datasets from Brazil, Greece, and Denmark, but like IP, MIP can be slow with complex problem instances.

Metaheuristic algorithms, such as SA, VNS, GDA, and HSA, offer alternative approaches by exploring the solution space in different ways. SA is a probabilistic method that searches for solutions while accepting worse ones with decreasing probability to avoid local optima. Although it is simple and easy to implement, SA tends to converge slowly and may not find the global optimum unless hybridized with other methods like ILS. VNS improves solutions by exploring multiple neighbourhood structures and escaping local optima. It is highly effective for smaller datasets but may falter with larger, more complex problems. GDA employs a threshold mechanism that accepts worse solutions to escape local minima and has been successfully applied to datasets like those from Tanzania, where it outperformed basic versions of the algorithm. However, GDA is often surpassed by other methods like SA. Meanwhile, HSA, inspired by the improvisation process of musicians, iteratively improves solutions by combining a memory-based process. HSA is particularly effective when hybridized

with SA, solving HSTP datasets with competitive results, though it requires careful parameter tuning to function optimally.

Matheuristics algorithms such as the Fix-and-Optimize Heuristic offer a combination of mathematical optimization and heuristic search methods. This approach has been applied to Brazilian school datasets and has shown improved performance over general-purpose solvers by employing specific decompositions, such as class, teacher, and day. However, its application is more limited in flexibility when compared to other approaches.

Heuristic and hyper-heuristic algorithms also play a key role in solving HSTP. GA, for example, mimic biological evolution through selection, crossover, and mutation. They have been successfully applied to datasets from South Africa, but their effectiveness can vary depending on the problem instance and may suffer from premature convergence. SI techniques, such as PSO, leverage the collective behavior of swarms to explore the solution space. SI algorithms have demonstrated exceptional performance in solving Greek datasets, but they tend to be sensitive to parameter settings. Hyper-heuristics combine multiple low-level heuristics to optimize solutions. They have been effective in generating best-known solutions for various datasets, though they require significant computational resources, especially for complex problems.

The selection of both HSA and GDA is justified by their complementary strengths. HSA's ability to handle complex datasets through a memory-based process, alongside GDA's capacity for escaping local optima through a threshold mechanism, ensures a

balanced approach to optimizing the timetabling problem. When combined as explained above, these methods provide a robust and flexible solution for handling the intricacies of scheduling in high schools, making them well-suited for large and complex datasets.

2.4.3 Neighbourhood Structures

In section 2.4.2 (Improvement Approaches), each approach required several neighbourhood structures **(**NS) to explore the solution space and find new, better solutions. These NS consist of move types and random selections. As the KHE had been used by the previous studies, the total 12 of NS utilized in KHE will be highlighted. The following lists all the NS:

1. **Swap Timeslots (**MeetsSwap**):** the day and timeslot of two meets are swapped. For example, the day and timeslot of meet named English on Monday (timeslot 3) and meet named Arabic at Friday (timeslot 6) are swapped. After the swap, Monday (timeslot 3) contains meet of Arabic, and Friday (timeslot 6) contains meet of English.
2. **Move Timeslot (**MeetsMoveTime**):** this move shifts the day and timeslot of meet to other days and timeslots that are available. As an example, the meet of Physic on Friday (timeslot 2) is moved to Monday (timeslot 3). After the move, the meet of Physic is scheduled on Monday (timeslot 3).
3. **Swap Resources (**TasksSwap**):** this task refers to the type of resources such as teacher, class, and room. This swap works with two resources. Two conditions are required, which are the two resources, such as teacher with only teachers, must be matched. For instance, the teacher named Teacher_01 of meet

English_02 is swapped with teacher named Teacher_03 of meet English_04. After the swap, the meet of English_02 is taught by Teacher_03, and the meet of English_04 is taught by Teacher_01.

4. **Move Resources (**TasksMoveResource**):** this task moves the available resources from instance to the solution, and unassigns the resources that are already inside the solution. For example, teacher Nour (which is already assigned in the solution) is moved out and replaced by teacher Ali, where teacher Ali is obtained from the instance.
5. **Block Meet Swap (**BlockMeetsSwap**):** block swapping is the same as ordinary swapping, except that it treats adjacent two or more same meets as one block. In Figure 2.6, when swapping a meet of one duration (Hist1) assigned with a meet of two durations (Arab2) assigned on Wednesday, Block Meet Swap swaps the adjacent of Arab2 meets with the Hist1 meet.

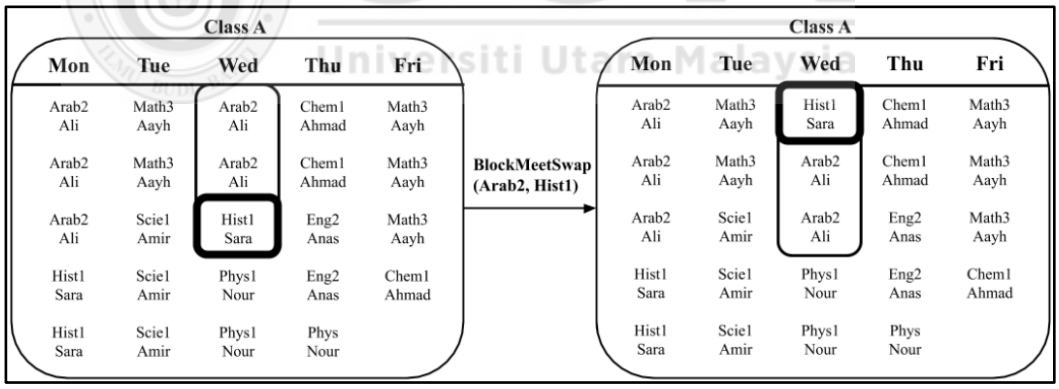


Figure 2.7. The Block Meet Swap neighbourhood.

6. **Kempe and Ejecting Moves**: in this particular approach, a set of techniques are employed to eliminate conflicts in an efficient and effective manner. The actions include unassigning, referred to as "ejecting," and relocating, referred to

as "Kempe." The ultimate goal of all these moves is to achieve the desired solution. Specifically, five specific types of moves are utilized to reach this goal:

a) **Kempe Meet Move (**KempeMeetMove**):** it moves meetings around in order to avoid conflicts. Figure 2.7 shows Wed2 of Arab1 (Class A and Class B) that is clashes as the teacher is same person which is Ali. Using this move, Wed2 of Arab1 in Class B will be swapped with Thu1 of Scie1 in the same class. This move removes the clashes.

b) **Kempe Meet Move Time (**KempeMeetMoveTime**):** it moves meetings around with one timeslot (from the solution space) and time (obtained from the instance). Figure 2.8 shows Wed2 of Arab1 (Class A and Class B) that is clashes as the teacher is same person which is Ali. Using this move, Scie1 in Thu1 will be moved to Wed2 (obtained from the instance) and Arab2 in Wed2 will be moved to the Thu1 (from the solution space). This move removes the clashes.

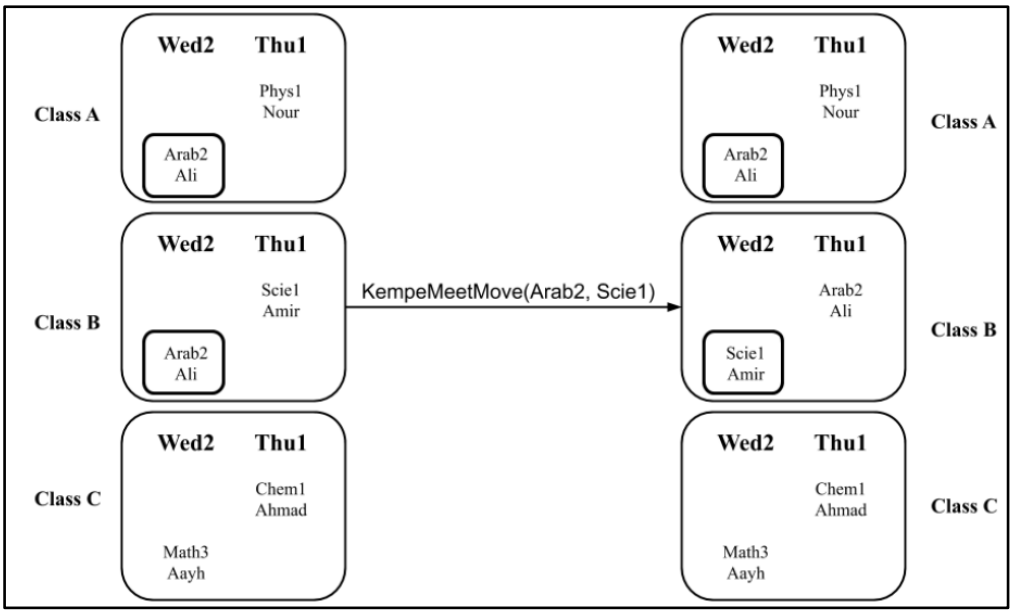


Figure 2.8. The Kempe Meet Move Neighbourhood.

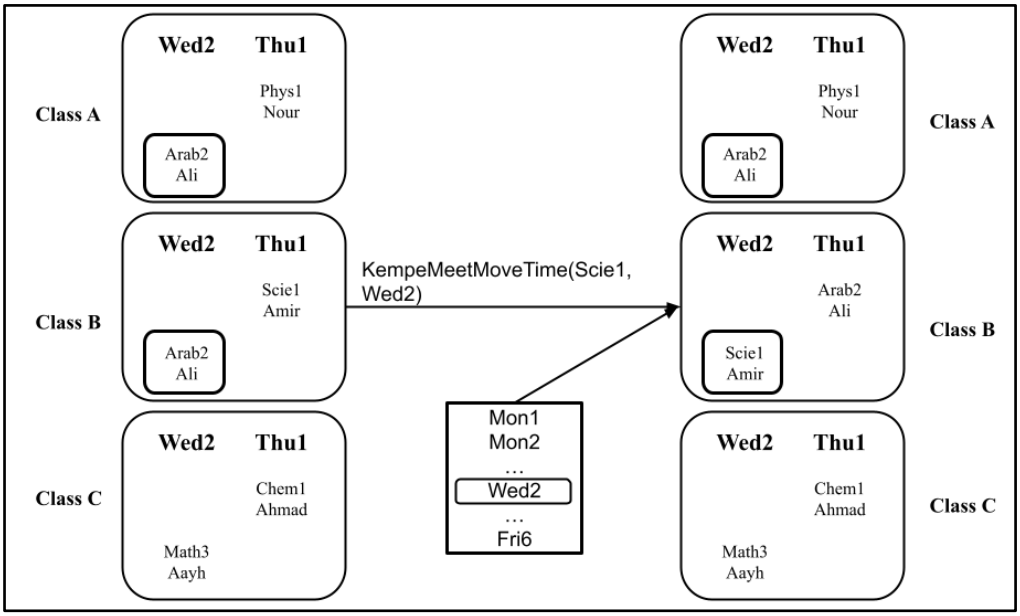


Figure 2.9. The Kempe Meet Move Time Neighbourhood

c) **Resource Ejecting Move (**ResourceEjectingMove**):** this transfers the resources (teacher, subject, room, or student) in the solution back to the instance, unassigning it from all clashing resources. The other resources (teacher, subject, room, or student) from the instance are assigned to the solution. This move ignores the matching, merely fails on preassigned times, and unassigns the clashing resources. Figure 2.9 shows that teacher Ali was unassigned from the timetable and teacher Kamal (obtained from instance) was assigned to all teacher Ali's slots.

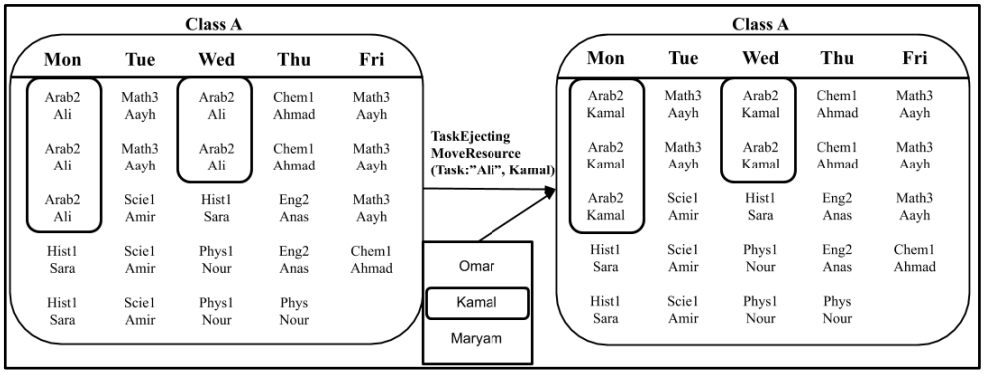


Figure 2.10. The Task Ejecting Move Resource Neighbourhood.

d) **Ejecting Kempe Meet Move (**EjectingKempeMeetMove**):** the ejecting meet move is a version of the Kempe meet move. This starts by moving the meet to the targeted meet at offset, then detects the meets that need to be moved back in the other direction, just like Kempe meet moves, but instead of moving them, it unassigns and stops them.

e) **Ejecting Kempe Meet Move Time (**EjectingKempeMeetMoveTime**):** this move is the same as the Ejecting Kempe Meet Move but operated with one timeslot (from the solution space) and time (obtained from the instance).

7. **Resource Pair Reassign (**ResourcePairReassign**):** A potential strategy for resolving resource assignment conflicts is to temporarily unassign all tasks that have been assigned to two conflicting resources, and subsequently attempt to re-allocate those tasks to the same two resources in a more optimal manner.
8. **Node Meet Swap (**NodeMeetSwap**):** Teachers and students attend one set of meetings (slots or nodes), which are then switched with another set of meetings at different times (slots or node).

2.4.4 The Best Methodologies Results Using XHSTT Datasets

As reviewed in Sections 2.4.2.1, 2.4.2.2, 2.4.2.3 and 2.4.2.4, a majority of the approaches utilized the XHSTT data set. Therefore, this section presents a summary of the results obtained by all the methodologies that solve HSTP for the XHSTT data instances only. Table 2.14 illustrates the best results of the XHSTT data instances based on metaheuristic methodologies, while Table 2.15 displays the best results based on mathematical optimization and matheuristic methods. For hyper-heuristic algorithms, the best results produced was only from Kheiri & Keedwell (2017).

Table 2.14 shows the top results obtained by employing metaheuristic techniques on XHSTT datasets from various countries compared to lower bound (optimal solution or the best possible solution that can be achieved) and initial solution (produced by KHE algorithm). The table indicates that the VNS method yielded over 50% of the best results for small, medium, and large datasets, outperforming other methods. Additionally, the VNS method generated four optimal solutions. The SA with ILS method performed well, coming in second place, producing good results on four small datasets, which include BrazilInstance3, FI-WP-06, UniversityInstance5, and ZL-LW-09. The SA with ILS generated three optimal solutions. The HSA-SA method came in third place, performing well on four datasets: BrazilInstance5, BrazilInstance7, SecondarySchool2 and ElementarySchool. These datasets include two small and two medium datasets. The HSA-SA method generated two optimal solutions. The SGVNS method performed well and took fourth place, producing good results on one small dataset (BR-SM-00) and two large datasets (UK-SP-06 and NL-KP-05). The SA-sf-LAHC method generated one optimal solution (UniversityInstance3). Additionally, the basic SA, RVNS, LAHC, SA-LAHC, and GVNS methods found one best solution for

the ArtificialSchool, Preveza2008, Aigio2010, Kottenpark2008, and GEPRO data instances, respectively.

Table 2.15 presents the best results obtained through the use of mathematical optimisation and matheuristic methods using XHSTT data instances compared to lower bound and upper bound (the maximum cost of possible solution that can be achieved). The majority of the results show that the lower bound has been reached. Out of 39 datasets, 12 of the best results were found by Fonseca et al. (2017) using IP. Kristiansen et al. (2015b) that proposed MIP methods produced 8 of the best results. The IP with two-stage decomposition model by Sørensen & Dahms (2014) generated one dataset (BR-SM-00) that equals to lower bound. The fix-and-optimize approach utilizing matheuristic method has demonstrated successful best results of three instances, namely BR-SN-00, BrazilInstance5, and BrazilInstance7 (Dorneles et al., 2014). However, both mathematical optimisation and matheuristic methods had not been producing results for several instances such as VillageSchool, FI-PB-98, FI-WP-06, SecondarySchool2, ArtificialSchool, GR-H1-97, ZL-LW-09, Preveza2008, Aigio2010, KS-PR-11, Kottenpark2008, GEPRO and ZL-LW-09.

In conclusion, various methods such as metaheuristic, mathematical optimisation, matheuristic and hyper-heuristic methods have been used to solve HSTP problem and majority of the results achieved the lower bound. Metaheuristics draw inspiration from natural phenomena, while exact or heuristic methods resemble the human approach to HSTP optimization.

Table 2.14

Best-known Result of XHSTT Data Instances by Metaheuristic Methodologies

Instance	Lower Bound	KHE	Results	Techniques Used
AU-BG-98	0.00000	3.00608	2.00398	VNS (Fonseca et al., 2016b)
AU-SA-96	0.00000	4.00022	4.00021	VNS (Fonseca et al., 2016b)
AU-TE-99	0.00020	2.00152	1.00036	VNS (Fonseca et al., 2016b)
BR-SA-00	0.00005	0.00031	0.00022	VNS (Fonseca et al., 2016b)
BR-SM-00	0.00051	0.00112	0.00090	SGVNS (Teixeira et al., 2019)
BR-SN-00	0.00035	0.00113	0.00104	VNS (Fonseca et al., 2016b)
BrazilInstance5	0.00019	12.09999	0.00148	HSA-SA (Shambour et al., 2013)
BrazilInstance3	0.00024	3.00240	≃0.00127	SA-ILS (Fonseca et al., 2016a)
BrazilInstance7	0.00040	22.09999	0.00234	HSA-SA (Shambour et al., 2013)
BrazilInstance1	0.00041	-	-	-
VillageSchool	0.00010	0.00657	0.00415	VNS (Fonseca et al., 2016b)
DK-FG-12	0.00412	0.03411	0.01669	VNS (Fonseca et al., 2016b)
DK-HG-12	-	12.04689	12.03371	VNS (Fonseca et al., 2016b)
DK-VG-09	2.00000	2.04691	2.02765	VNS (Fonseca et al., 2016b)
ES-SS-08	-	0.00657	0.00415	VNS (Fonseca et al., 2016b)
FI-PB-98	-	0.00000	0.00000	VNS (Fonseca et al., 2016b)
FI-WP-06	-	0.00001	0.00001	SA+ILS (Teixeira et al., 2019)
FI-MP-06	0.00077	0.00102	0.00084	VNS (Fonseca et al., 2016b)
SecondarySchool2	-	2.09999	0.00000	HSA-SA (Shambour et al., 2013)
ElementarySchool	0.00003	10.00306	0.00003	HSA-SA (Shambour et al., 2013)
ArtificialSchool	-	20.00014	19.00012	SA (Brito et al., 2012)
GR-H1-97	-	0.00000	0.00000	VNS (Fonseca et al., 2016b)
GR-P3-10	-	0.00002	0.00000	VNS (Fonseca et al., 2016b)
GR-PA-08	0.00003	0.00011	0.00003	VNS (Fonseca et al., 2016b)
UniversityInstance3	0.00005	0.00030	≃0.00005	SA-sf-LAHC (Fonseca et al., 2016)
UniversityInstance5	-	17.00044	≃0.00000	SA-ILS (Fonseca et al., 2016a)
Preveza2008	-	13.00603	0.00033	RVNS (Brito et al., 2012)
Aigio2010	-	14.00757	0.00011	LAHC (Fonseca et al., 2016)
IT-I4-96	0.00027	0.00046	0.00042	VNS (Fonseca et al., 2016b)
KS-PR-11	-	0.00017	0.00004	VNS (Fonseca et al., 2016b)
NL-KP-03	-	0.01371	0.01151	VNS (Fonseca et al., 2016b)
NL-KP-05	0.00089	0. 01078	0.00877	SGVNS (Teixeira et al., 2019)
NL-KP-09	-	10.05125	8.08370	VNS (Fonseca et al., 2016b)
Kottenpark2008	0.02795	63.140083	≃10.57476	SA-LAHC (Fonseca et al., 2016)
GEPRO	1.00000	1.00566	1.00434	GVNS (Teixeira et al., 2019)
UK-SP-06	-	0.18444	0.12466	SGVNS (Teixeira et al., 2019)
US-WS-09	0.00101	0.00532	0.00124	VNS (Fonseca et al., 2016b)
ZL-LW-09	-	0.00058	0.00000	SA-ILS (Fonseca et al., 2016a)
ZA-WD-09	-	16.00000	3.00000	VNS (Fonseca et al., 2016b)

Note: *Dash (-)* denotes that there are no results provided by the author.

Table 2.15

Results of XHSTT Instances by Mathematical Optimisation and Matheuristic Methods

Instance	Lower Bound	Upper Bound	Results	Techniques Used
AU-BG-98	0.00000	0.00129	0.00129	IP (Fonseca et al., 2017)
AU-SA-96	0.00000	0.00000	0.00000	IP (Fonseca et al., 2017)
AU-TE-99	0.00020	0.00020	0.00020	IP (Fonseca et al., 2017)
BR-SA-00	0.00005	-	0.00005	MIP (Kristiansen et al., 2015b)
BR-SM-00	0.00051	-	0.00051	Two-Stage Decomposition (Sørensen & Dahms, 2014)
BR-SN-00	0.00035	-	0.00035	fix-and-optimize (Dorneles et al., 2014)
BrazilInstance5	0.00019	-	0.00019	fix-and-optimize (Dorneles et al., 2014)
BrazilInstance3	0.00024	-	0.00024	MIP (Kristiansen et al., 2015b)
BrazilInstance7	0.00040	-	0.00053	fix-and-optimize (Dorneles et al., 2014)
BrazilInstance1	0.00041	-	0.00041	MIP (Kristiansen et al., 2015b)
VillageSchool	0.00010	-	-	-
DK-FG-12	0.00412	0.01300	0.01263	IP (Fonseca et al., 2017)
DK-HG-12	7.00000	12.02356	12.02330	IP (Fonseca et al., 2017)
DK-VG-09	2.00000	2.02329	2.02323	IP (Fonseca et al., 2017)
ES-SS-08	0.00334	0.00657	0.00335	MIP (Kristiansen et al., 2015b)
FI-PB-98	-	-	-	-
FI-WP-06	-	-	-	-
FI-MP-06	0.00077	-	0.00083	MIP (Kristiansen et al., 2015b)
SecondarySchool2	-	-	-	-
ElementarySchool	0.00003	-	0.00003	MIP (Kristiansen et al., 2015b)
ArtificialSchool	-	-	-	-
GR-H1-97	-	-	-	-
GR-P3-10	-	-	-	-
GR-PA-08	0.00003	-	0.00003	MIP (Kristiansen et al., 2015b)
UniversityInstance3	0.00005	-	0.00005	MIP (Kristiansen et al., 2015b)
UniversityInstance5	-	-	-	-
Preveza2008	-	-	-	-
Aigio2010	-	-	-	-
IT-I4-96	0.00027	-	0.00027	IP (Fonseca et al., 2017)
KS-PR-11	-	-	-	-
NL-KP-03	0.00000	0.00119	0.00119	IP (Fonseca et al., 2017)
NL-KP-05	0.00089	0. 00433	0.00425	IP (Fonseca et al., 2017)
NL-KP-09	0.00180	0.01620	0.01620	IP (Fonseca et al., 2017)
Kottenpark2008	-	-	-	-
GEPRO	-	-	-	-
UK-SP-06	0.0000	5.04014	5.04014	IP (Fonseca et al., 2017)
US-WS-09	0.00101	0.00103	0.00103	IP (Fonseca et al., 2017)
ZL-LW-09	-	-	-	-
ZA-WD-09	-	-	0.00000	MIP (Kristiansen et al., 2015b)

Note: *Dash (-)* denotes that there are no results provided by the author.

Metaheuristics are broadly categorized into two groups: population-based and local search algorithms. This study employs both types of metaheuristics to generate diverse solutions (Gashgari et al., 2021). Comparison between Swarm Intelligence versus Evolutionary Algorithm as outlined in Table 2.16 shows that HSA is less complex in concept when compared to intelligence swarm algorithms such as PSO, CSO, and AFS (Anwar et al., 2013; Du & Swamy, 2016). Unlike these algorithms, HSA is not sensitive to the initial parameter settings and does not require additional parameters for velocity values during the search space (Du & Swamy, 2016). Moreover, HSA utilizes a probabilistic gradient mechanism to progressively converge towards a better solution during each iteration, making it suitable for both continuous and discrete optimization problems (Jordehi & Jasni, 2015). In contrast, PSO, CSO, and AFS are weaker in discrete optimization problems due to their mathematical calculations being linked to continuous numbers in the search space (Du & Swamy, 2016). Furthermore, the iterations of PSO, CSO, and AFS require calculus involving derivatives and integrals, making the algorithm more complicated and difficult to apply in domains with multiple constraints (Du & Swamy, 2016). Therefore, HSA is preferred over PSO, CSO, and AFS in situations where small changes may occur during optimization, as HSA can adapt more effectively to these changes. Next section describes the review of HSA.

Table 2.16

Comparison Swarm Intelligence versus Evolutionary Algorithm

Facts	Swarm Intelligence			Evolutionary Algorithm	
	PSO	**CSO**	**AFS**	**HSA**	**GA**
Minimal in concept (Anwar et al., 2013; Du & Swamy, 2016)	No	No	No	Yes	Yes
Probabilistic gradient (Du & Swamy, 2016; Rezaee Jordehi & Jasni, 2015)	No	No	No	Yes	No
Calculus required (Du & Swamy, 2016)	Yes	Yes	Yes	No	No

2.5 Harmony Search Algorithm (HSA)

HSA is categorized as an evolutionary algorithm, which mimics the musical improvisation process of jazz musicians, aims to find better solutions in few iterations. Each instrument of musical defines the aesthetic quality, which is called as pitch, that same as with the objective function which defines the decision variables quality. In the improvisation process of the music, combination of sounds considered pleasing from an aesthetic point of view produce music harmony (one solution). If the entire pitches produce a great harmony, each music player's memories experiences and the possibility of producing a great harmony is improved with time (Abdulkhaleq et al., 2022; Geem et al., 2001). The similar matter in optimization, the initial solution is created at random from decision variables surrounded by the possible range. If these decision variable's objective function is great to achieve a favorable solution, then the possibility to produce a good quality solution is improved next time (Geem et al., 2001).

Table 2.17 shows the comparison analogy between numerical optimization context and concepts of harmony search (musical performance). The harmony memory (matrix)

contains harmonies (solution vectors) played by musicians (decision variables). Improvising (generating) a new harmony (new solution vector) is by practicing (iteration) which take the good notes (values of the variables) previously played by the musician (decision variable) as well as take the pitch (value) to be modified to a value of neighboring. The new harmony (new solution vector) is then evaluated against audio aesthetic standard (objective function). Finally, pleasing harmony (good solution) replaces the worst harmony (solution vector) in the harmony memory (matrix) (Du & Swamy, 2016). The HSA consists of several stages (Geem, 2010) as shown in Figure 2.12, which includes **1) problem formulation:** numerous problems can be formulated as real-world optimization problem (such as timetabling problem) that involves constraints either hard or soft while minimizing or maximizing a specified objective function.

Table 2.17

Comparison Analogy between Numerical Optimization Context and Concepts of Harmony Search

Concepts of Harmony Search (Musical Performance)	**Numerical Optimization Context**
Musicians	Decision Variables
Notes	Values of The Variables
Harmony	Solution Vector
Improvisation	Generation
Pitch	Value
Audio Aesthetic Standard	Objective Function
Practice	Iteration
Pleasing Harmony	Good Solution
Harmony Memory	Solution Vector(s)/Matrix

2) Algorithm parameter setting: five parameters are used, which are harmony memory consideration rate (HMCR), random consideration rate (RCR or 1-HMCR) as indirect parameters, pitch adjustment rate (PAR), maximum improvisation (MI), and

harmony memory size (HMS). **3) Random tuning for memory initialization:** process of creating the initial population, that is also known as harmony memory (HM) based on the number of HMS. **4) Harmony improvisation:** iteration of HSA in which a solution is randomly chosen from the HM and improved through random selection of several operators such as memory consideration (MC), random consideration (RC) and pitch adjustment (PA) applied with small changes using NS such as, swap or move as described in section 2.4.3. **5) Memory update**: replacing the new harmony or solution, if the objective function is improved than the worse solution or harmony available in the HM. **6) Performing termination**: stop the iteration of HSA when reaching the value of MI. **7) Cadenza:** demonstrates the best solution in HM.

The calculation of original HSA's pitch adjustment rate (PAR) is shown in the Equation 2.7 which calculates the difference between the maximum and minimum values of PAR, scales it by t, and then divides by MI. As t increases from 0 to MI, $PAR(t)$ will linearly interpolate from PAR_{min} to PAR_{max}. In simpler terms, the formula is used to find the value of PAR at any given iteration t between 0 and MI, scaling linearly between PAR_{min} and PAR_{max}.

$$PAR(t) = PAR_{min} + (PAR_{\max} - PAR_{min}) \times \frac{t}{MI} \tag{2.7}$$

Where

PAR - pitch adjusting rate for each generation

PAR_{min} - minimum pitch adjusting rate

PAR_{max} - maximum pitch adjusting rate

MI - maximum number of solution vector generations

t - generation number

The basic HSA has been enhanced through modification and hybridization with other algorithms. Modified HSA is more popular among researchers compared to hybrid versions (Abualigah et al., 2020). This is due to the weaknesses of HSA that have difficulty in regulating, adjusting, or modifying its parameters to enable it to work with the solution space and problem. In addition, the fixed parameters of basic HSA can impact the quality of the solution obtained (Du & Swamy, 2016). However, the enhancement of HSA based on hybridization is also recommended by researchers so that the proposed techniques of HSA could be included with any other methods. The hybridization will enable the parameters of HSA to be tuned dynamically to handle extra-complex optimization problems (Abualigah et al., 2020). The following subsections explain modification and hybridization techniques on the previous enhancement of HSA proposed in the literature.

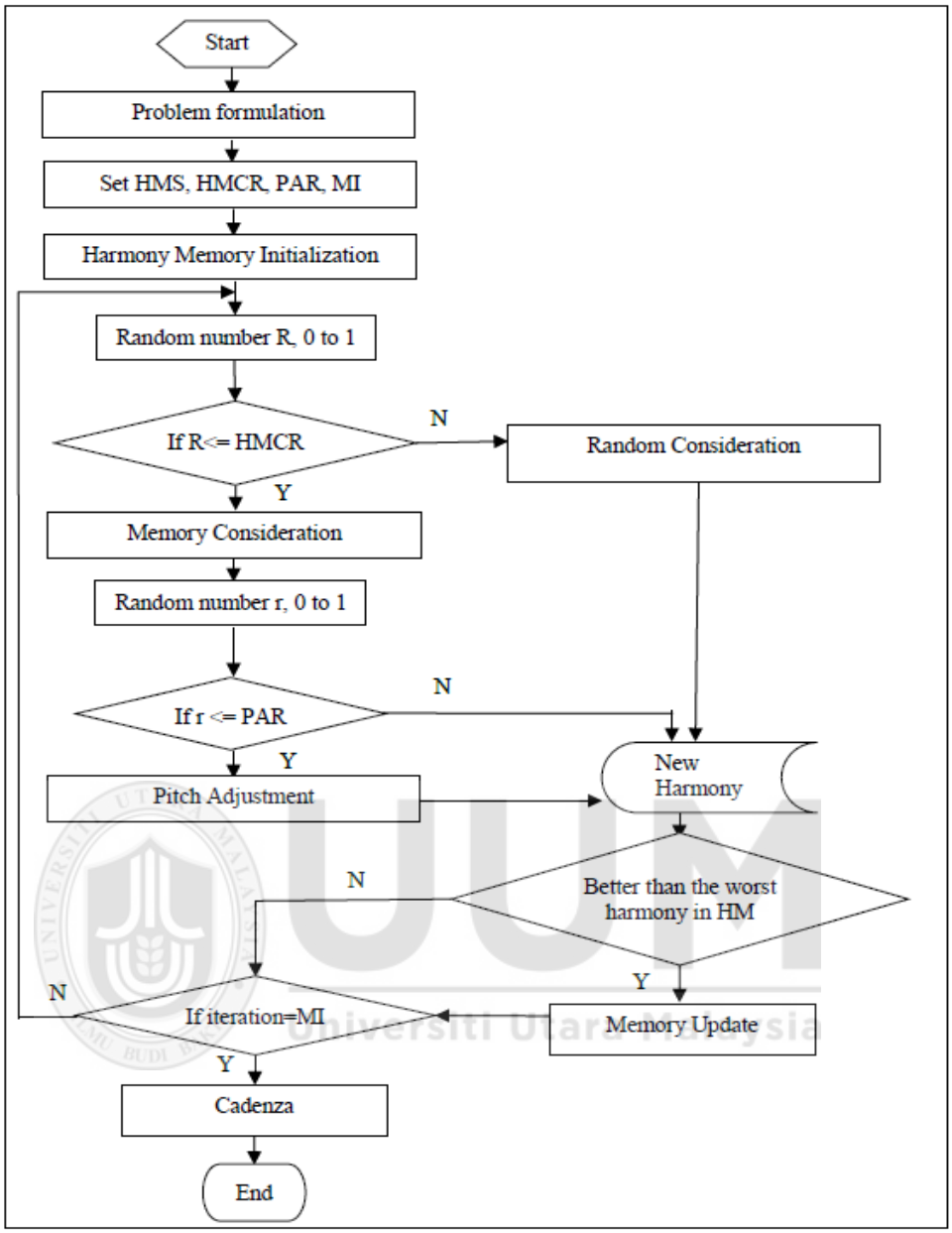


Figure 2.11. Harmony Search Algorithm Flowchart (Wahid, 2017)

2.5.1 The Modification Variants of HSA

One of the primary challenges in using HSA is the lack of correlation between the search space and the problem at hand, as fixed parameters can cause the search space to become disconnected from the problem and converge rapidly (Wang et al., 2019). To address this limitation, researchers have shifted their focus towards incorporating dynamic parameters to improve the effectiveness of fixed HSA and minimize the gap

between the search space and the problem. These approaches seek to enhance the HSA's adaptability and responsiveness to the unique requirements of different problems. The improved variants of HSA are focused on the algorithmic itself by regulate or adjust the parameters to be able to work with solution space and the problem (Du & Swamy, 2016; Wang & Huang, 2010).

The parameters of HSA had been discussed by numerous researchers in dynamically adjusting it during the improvement of search space in various optimizations problems. Mahdavi et al. (2007b) applied PAR that changed dynamically with generation number as shown in Equation 2.8. This formula was implemented with several problems, such as minimization of spring weight, pressure vessel design, and constrained function V which produced better results compared with standard HSA and other heuristics.

$$PAR(gn) = PAR_{min} + \frac{(PAR_{\max} - PAR_{min})}{NI} \times gn \tag{2.8}$$

Where

PAR - pitch adjusting rate for each generation

PAR_{min} - minimum pitch adjusting rate

PAR_{max} - maximum pitch adjusting rate

NI - number of solution vector generations

gn - generation number.

Omran & Mahdavi (2008) modified HSA based on concept of SI in PSO in which the global-best harmony search (GHS) was introduced. GHS modified the PA step of the HSA such that the new harmony can mimic the best harmony in the HM. Their

study evaluated the performance of the proposed approach against ten benchmark numerical optimization problems and found that it generally outperformed other methods. The investigation focused on examining the impact of noise on the performance of different variations of the HSA. The empirical results indicated that the GHS algorithm yielded superior results when applied to high-dimensional problems.

Wang & Huang (2010) used PAR that decreased with time to prevent overshooting and oscillation. In the study, the parameters of HSA were routinely adjusted to match to its self-consciousness by selecting suitable parameter values on the basis of self-adaptive mechanism. This mechanism was applied to different four common numerical optimization problems, such as *Sphere, Rosenbrock,* and *Ackley*. The numerical results demonstrate, with 99% confidence, that their approach surpasses the performance of basic harmony search, enhanced harmony search, and global-best harmony search in all four common numerical optimization problems.

Chen et al. (2012) adjusted the PAR dynamically with the generation number solving numerical optimization problems as shown in Equation 2.9. The results prove that the dynamic HSA is superior to the standard HSA in conditions of solution quality and performance.

$$PAR(t) = PAR_{min} + (PAR_{\max} - PAR_{min}) \times \exp(-k\ t/NI) \quad (2.9)$$

Where

PAR - pitch adjusting rate for each generation

PAR_{min} - minimum pitch adjusting rate

PAR_{max} - maximum pitch adjusting rate

NI - number of solution vector generations

t - generation number

k - constant value

Peraza et al. (2016) changed dynamically values of MCR and PAR parameters that used as inputs to a fuzzy systems that solve numerical optimization problems such as sphere, Rosenbrock, and Rastrigin. The MCR and PAR formula are defined by Equations 2.10 and 2.11, respectively. The dynamic parameters demonstrate better results compared to the standard HSA.

$$MCR = \frac{\sum_{i=1}^{r_{MCR}} \mu_i^{MCR}(MCR_{1i})}{\sum_{i=1}^{r_{MCR}} \mu_i^{MCR}} \tag{2.10}$$

$$PAR = \frac{\sum_{i=1}^{r_{PAR}} \mu_i^{PAR}(PAR_{1i})}{\sum_{i=1}^{r_{PAR}} \mu_i^{PAR}} \tag{2.11}$$

Where

MCR - memory consideration rate

r_{MCR} - the number of rules of the fuzzy system corresponding to MCR

MCR_{1i} - the output result for rule i corresponding to MCR

μ_i^{MCR} - is the membership function of rule i corresponding to MCR

PAR - pitch adjusting rate

r_{PAR} - the number of rules of the fuzzy system corresponding to PAR

PAR_{1i} - the output result for rule i corresponding to PAR

μ_i^{PAR} - is the membership function of rule i corresponding to PAR

Li et al. (2017) applied dynamic PAR where a low PAR value was used in the beginning stages to steer the search towards promising areas. As the search progressed, a higher PAR value was utilized to prevent getting stuck in local optimums. The formula outlined in Equation 2.12 serves as the foundation for resolving assembly sequence planning problems. This formula has been demonstrated to be effective in improving the rate of convergence and augmenting the likelihood of obtaining optimal solutions.

$$PAR(t) = (PAR_{\max\_} PAR_{min}) \frac{\arctan t}{\frac{\pi}{2}} + PAR_{min} \quad (2.12)$$

Where

PAR - pitch adjusting rate for each generation

PAR_{min} - minimum pitch adjusting rate

PAR_{max} - maximum pitch adjusting rate

t - number of generations

$arctan$ – function along with time.

Wang et al. (2019) proposed improved differential HSA (IDHS) to solve numerical function optimization problems. In their proposed algorithm, the harmony consideration phase (MC and PA operators) was set based on Equation 2.12 and 2.13, respectively. In Equation 2.13, $HMCR_{min}$ was increased during the process using $(HMCR_{max} - HMCR_{min}) \times it/NI$, while in Equation 2.14, the PAR_{max} value was dynamically decreased using $(PAR_{max} - PAR_{min}) \times it/NI$ to balance the algorithm's exploration and exploitation abilities.

$$HMCR(it) = HMCR_{min} + (HMCR_{max} - HMCR_{min}) \times \frac{it}{NI} \quad (2.13)$$

$$PAR(it) = PAR_{max} - (PAR_{max} - PAR_{min}) \times \frac{it}{NI} \quad (2.14)$$

Where

it - is the current iteration.

NI - is the total number of iterations.

$HMCR_{min}$ - minimum harmony random consideration rate

$HMCR_{max}$ - maximum harmony random consideration rate

PAR_{min} - minimum pitch adjusting rate

PAR_{max} - maximum pitch adjusting rate

Hasanipanah et al. (2022) proposed adaptive dynamical harmony search algorithm (ADHS) for accurate prediction of blast-induced flyrock. It involved making adjustments to the harmony elements based on the information gathered from the harmony memory at each iteration. This involved dynamically updating the position of the harmony elements using two random adjustment procedures. The first procedure involved adjusting the harmony elements using the maximum and minimum values for each random variable in harmony memory, with a consideration given to a dynamic harmony memory rate (HMCR), as shown in Equation 2.15. On the other hand, the variable's best memory was optimized by means of an adjusting step size that is determined by taking into account the information collected from the harmony elements using the second random procedure. This involved utilizing the dynamical pitch adjusting rate (PAR) to fine-tune the best harmony memory, as shown in Equation 2.16.

$$HMCR(k) = 0.95 + 0.1 \times \sqrt{\frac{k}{NI} - (\frac{k}{NI})^2} \quad (2.15)$$

$$PAR(k) = 0.3 + 0.6 \times \left[1 - \sqrt{1 - \frac{k}{NI}}\right] \quad (2.16)$$

Where

k - is the current iteration.

NI - is the total number of iterations.

The improvements and modifications to HSA discussed above provide a range of approaches aimed at enhancing its effectiveness in different optimization problems. One of the major modifications involves dynamically adjusting parameters such as the PAR and HMCR, which significantly improve the balance between exploration and exploitation in the search space. These dynamic adjustments allow HSA to adapt to different problem domains and reduce the risk of premature convergence, leading to superior performance in high-dimensional optimization problems. For instance, Mahdavi et al. (2007a) and Chen et al. (2012) demonstrated that adjusting the PAR dynamically with the generation number improves both the solution quality and performance, as evidenced by better results in problems like spring weight minimization and constrained function V. Similarly, the introduction of the GHS by Omran & Mahdavi (2008), which integrates the concept of PSO, further improves the search process by allowing the new harmony to mimic the best harmony in the memory, producing superior results in benchmark problems. However, these improvements often come at the cost of increased computational complexity, requiring more fine-tuning of parameters and continuous evaluation of solutions, particularly in real-world problems

with complex landscapes. Moreover, approaches like Peraza et al. (2016) fuzzy logic-based modifications to dynamically adjust MCR and PAR show promising results for solving numerical optimization problems like Sphere and Rosenbrock, but they add further complexity to the algorithm due to the introduction of fuzzy logic systems that require careful design of membership functions and rules. This approach has been shown to outperform standard HSA, but the added computational overhead and tuning challenges make it less accessible for general applications. Wang & Huang (2010) introduced a method to prevent overshooting and oscillation by dynamically adjusting the PAR in their improved HSA variant, demonstrating superior performance in common optimization problems such as Sphere, Rosenbrock, and Ackley. However, while these dynamic adjustments improve adaptability and convergence rates, they often increase the computational burden and require careful selection of initial parameters. In practical applications, the suitability of each method depends on the specific problem and computational resources available. For example, the improved differential HSA proposed by Wang et al. (2019) provides a balance between exploration and exploitation by dynamically adjusting the HMCR and PAR, which helps prevent overshooting and oscillation, showing better performance across a range of numerical function optimization problems. Yet, this approach, like many others, demands significant computational resources and parameter tuning. Similarly, the ADHS proposed by Hasanipanah et al. (2022) demonstrated strong predictive performance in real-world problems such as blast-induced flyrock prediction, but this continuous dynamic adaptation introduces a level of complexity that can be difficult to generalize to other domains. Despite the advantages of these modifications in enhancing the HSA's effectiveness in various optimization problems, they all share the

challenge of added computational complexity and the need for fine-tuning, which can limit their practicality in broader contexts, Table 2.18 summarizes these approaches.

Table 2.18

Summary of Approaches to Improving Harmony Search Algorithm (HSA) with Results, Pros, and Cons

Approach	**Results**	**Pros**	**Cons**
Dynamic Parameter Adjustments	Improved convergence and adaptability	Enhances exploration and exploitation balance	Increased complexity and computational overhead
PAR Modifications	Superior results in specific problems (e.g., assembly sequence planning)	Avoids local optima and improves solution quality	Requires careful tuning and domain-specific parameter selection
Global-Best Harmony Search (GHS)	Superior performance in high-dimensional optimization problems	Incorporates features from PSO for better search performance	Computationally intensive and may struggle with complex landscapes
Fuzzy Logic-Based Modifications	Outperforms standard HSA in problems like Sphere and Rosenbrock	Flexible and adaptive adjustments through fuzzy systems	Adds complexity and computational overhead
Improved Differential HSA	Effective in balancing exploration and exploitation	Works well in a variety of optimization problems	Dependent on initial parameter settings and dynamic strategies
Adaptive Dynamical Harmony Search (ADHS)	Strong predictive performance in real-world problems	Continuously adapts to problem space	Difficult to generalize and increases computational demands

The adaptive HSA works that been described above, as summarized as shown in Table 2.19, have limitations which can be further improvised. Firstly, consideration of only one search space dimension using the position of iteration, which restricts the ability to adjust the HSA parameters during the search space and isolate it from the search space

of a specific solution. To overcome this, adaptive HSA parameters should consider two search space dimensions such as iteration position and solution number. Secondly, the previous adaptive HSA works (Chen et al., 2012; Hasanipanah et al., 2022; Li et al., 2017; Mahdavi et al., 2007b; Peraza et al., 2016; Wang & Huang, 2010; Wang et al., 2019) have not considered the behavioural status of cost solution (objective function), which limits the ability of HSA to track the best behavioural neighbourhood moves that can improve the solution. Thus, the adaptive HSA parameters should be adjusted based on the behavioural status of objective function obtained in solution. Thirdly, the previous adaptive HSA works (Chen et al., 2012; Hasanipanah et al., 2022; Li et al., 2017; Mahdavi et al., 2007b; Peraza et al., 2016; Wang & Huang, 2010; Wang et al., 2019) did not provide the linkage mechanism between the adaptive parameters and previous fixed parameters of HSA related to the same domain. This means that the adaptive HSA parameter rates should be adjusted based on linkage mechanism due to the decision made during the search space.

Table 2.19

Limitation Facts of Previous Adaptive HSA

References	Dimension of the search space based on		Changeable parameters based on behavioral status	Parameter linkages to basic HSA works	Parameters			
	Iteration position	Solution number			Iteration size	HMCR	RCR	PAR
An improved harmony search algorithm (Mahdavi et al., 2007a)	Yes	No	No	No	No	No	No	Yes
Self-adaptive harmony search algorithm (Wang & Huang, 2010)	Yes	No	No	No	No	No	No	Yes
Harmony search algorithm with dynamic control parameters (Chen et al., 2012)	Yes	No	No	No	No	No	No	Yes

A new fuzzy harmony search algorithm (Peraza et al., 2016)	Yes	No	No	No	No	Yes	No	Yes
A dynamic parameter controlled harmony search (Li et al., 2017)	Yes	No	No	No	No	No	No	Yes
Improved differential harmony search (Wang et al., 2019)	Yes	No	No	No	No	Yes	No	Yes
An ANN adaptive dynamical harmony search algorithm (Hasanipanah et al., 2022)	Yes	No	No	No	No	Yes	No	Yes

Fourthly, the previous works on adaptive HSA were focused only on the HMCR and PAR parameters, whereas the RCR and iteration size of space search are also important parameters. All HSA parameters should be incorporated in producing the adaptive HSA. Considering lack of adaptive parameter settings of HSA in previous studies, thus a new strategy on improvisation scheme for HSA is proposed which characterize the combination between iteration position and solution number, behavioural status of neighbourhood moves, and parameters linkages.

The new approach is able to link the past situation with current one which was inspired from the strategy of word repetition from the Holy Qur'an (Khan et al., 2018). The Holy Quran often reiterates specific themes, reinforcing its core messages for the reader. For instance, when discussing the concept of heaven, each subsequent reference builds on the previous one, enhancing the understanding of this concept with additional information.

Table 2.20 demonstrates the analogy between the Holy Qur'an concept and dynamic HSA parameters. The frequency position Ayah in Holy Qur'an (the iteration): This suggests that each iteration of the algorithm corresponds to a different position in the Qur'an, likely in the order of the verses (Ayahs). Surah (the solution): Surahs are the chapters of the Qur'an. The association here may suggest that each Surah offers a 'solution' or outcome, similar to how each run of an algorithm might offer a different result or solution. Based on these two concepts (iteration and solution), the searching for solution will be more better compared to one-dimensional search space (iteration) (Vitor, 2019). Verse/Ayah (Behavioral status): The verses (Ayahs) could be seen as defining the "state" of the reader at a given time. Similarly, the behavioral status might represent the state of the algorithm at a given iteration. Searching for behavioral diversity alone often more effectively leads to evolving solutions than does searching directly for high-quality solutions (Lehman, 2017). Repeated the word as heaven/earth/sky (the neighborhood-move): In optimization algorithms, a 'neighborhood-move' refers to a slight adjustment made to a current solution to create a new potential solution. This could correspond to repeated words altering the 'state' of the reader slightly. The adjective of the word that differs from one Verse/Ayah to another Verse/Ayah (the set of NS): NS might stand for Neighborhood Search, which involves searching the 'neighborhood' of a current solution for better solutions. Here, adjectives changing from one verse to another might correspond to 'moves' within the solution space. The reader's understanding that links the meanings from the previous adjective word to the current reading word adjective of the word (Parameter linkages (a mechanism)): The understanding of the reader might evolve as they progress through the Qur'an. Similarly, parameter linkages refer to how changes in one parameter may affect others, influencing the overall solution. One of the main steps to look at the

characteristics of nature-inspired algorithms is “search mechanisms” in order to enhance the search space (Yang, 2020). More adjectives found in the Holy Qur’an, the more the reader is guided to understand the meanings (Iteration size): This could be interpreted as a metaphor for how the amount of 'information' (in this case, adjectives in the Qur'an) corresponds to the depth of understanding. Similarly, more iterations in the algorithm may lead to better or more refined solutions. Based on that, the iteration size can adaptively adjust the search direction and search area (Xu et al., 2019).

Table 2.20

Comparison Analogy between the Holy Qur’an and Dynamic HSA

Holy Qur’an concept	**Properties of dynamic HSA**
The frequency position Ayah in Holy Qur’an	The iteration
Surah	The solution
Verse/Ayah	Behavioral status
Repeated the word as heaven/earth/sky	The neighborhood-move
The adjective of the word that differs from one Verse/Ayah to another Verse/Ayah	The set of NS
The reader’s understanding that links the meanings from the previous adjective word to the current reading word adjective of the word	Parameter linkages (a mechanism)
More adjectives found in the Holy Qur’an, the more the reader is guided to understand the meanings	Iteration size

2.5.2 The Hybridization Variants of HSA

The forms of hybridization that can be combined with the HSA (Blum & Roli, 2003) include:

1. Integrating elements from other metaheuristics: This involves incorporating strategies or techniques from other metaheuristics into the harmony search algorithm to enhance its performance.

2. Cooperative search: In this form, the harmony search algorithm is combined with other algorithms to exchange information and work together to find an optimal solution.
3. Integration of approximate and systematic methods: This form of hybridization involves combining the harmony search algorithm with systematic methods, such as mathematical programming or constraint programming, to achieve a more accurate solution.

On top of these three forms of hybridization, the theoretical foundation of metaheuristic optimization is based on two opposite criteria (exploration and exploitation) for the best solution (Koopialipoor & Noorbakhsh, 2020b). Figure 2.12 indicates the difference between population-based algorithms and single or local-based algorithm. It is shown in the figure that local search focuses on one factor (such as one solution, or particular thing in solution) to optimize the solution, whereas the population-based search (such as HSA) has diversity through population solution.

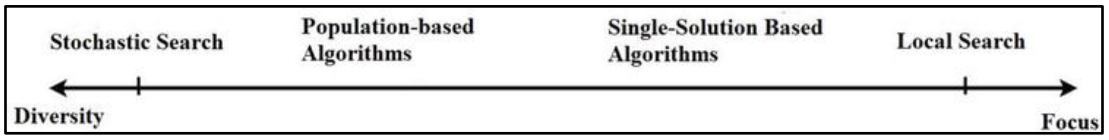


Figure 2.12. The Space of Metaheuristic Algorithm Design (Koopialipoor & Noorbakhsh, 2020b)

For the first form of hybridization, Al-Betar et al. (2012) combined the HSA with HC for solving CTP. This is a type of hybrid optimization technique, where two or more algorithms are combined to take advantage of the strengths of each algorithm and overcome their weaknesses. In this case, the HSA is used to generate a new harmony

solution, which is then fine-tuned using the HC method with a probability determined by the HC Rate (HCR). The hybridization was able to generate great results for the most challenging problem instances of university CTP. Wahid & Hussin (2017) hybridized HSA with GDA for solving the university curriculum-based CTP. GDA was hybridized with HSA in the RC operator as acceptance method. The results indicated that the quality of timetable generated was better than the timetable made using the existing software package. Assad & Deep (2018) proposed a hybrid algorithm that combined HSA and SA for solving the camera calibration problem. The temperature parameter of the SA was utilized to determine the probability of accepting an inferior solution, allowing for a balance between exploration and exploitation. The temperature was initially set high to favour exploration, but gradually decreased to focus on exploitation. The results from statistical tests indicate that the proposed hybrid algorithm was highly efficient. Gheisarnejad (2018) enhanced the performance of HSA by integrating it with the Cuckoo Optimization Algorithm (COA) and a fuzzy PID controller to solve the Load Frequency Control (LFC) problem. The egg-laying mechanism of COA was utilized as a local search strategy to refine the solutions during the evolution of HSA. To implement this mechanism, the egg-laying radius (ELR) of the candidate solutions accumulated in the harmony memory (HM) was first calculated, and then some local solutions, referred to as eggs in COA, were assigned to each of the HM solutions based on the corresponding ELR. The locally generated solutions were then added to the HM. Empirical results showed that this approach provided better quality results compared to previous reported strategies for load frequency control.

For the second form of hybridization, Shambour et al. (2013) integrated the HSA with SA to address HSTP. The proposed hybrid method involves performing an iteration of

HSA followed by the application of the SA algorithm as a refinement step, aimed at enhancing the current optimal solution. The empirical results demonstrated the effectiveness of the hybridized method, showing competitive and satisfactory outcomes.

For the third form of hybridization, Fesanghary et al. (2008) proposed a hybrid algorithm that combines the use of HSA to produce a nearly global search region utilizes Sequential Quadratic Programming (SQP) for local search. Upon reaching the established termination criteria of the HSA, the SQP local search is applied to vectors stored in the HM as a final refinement to determine the optimal solution. The effectiveness of this algorithm was demonstrated through its application to various engineering optimization problems and its results were compared to those produced by other traditional methods. The findings indicated that the hybrid approach was successful in obtaining high-quality solutions and efficient in terms of the number of fitness function evaluations required. Yildiz & Öztürk (2010) introduced an optimization method that combined the use of the HSA and Taguchi's method to address shape optimization problems. Taguchi's method was utilized to establish robust initial population levels of design parameters and minimize the impact of noise factors for improved initial HM. The challenge with a larger population is that it often converges and becomes stuck around certain solutions which may not be the optimal one. This issue was addressed through the integration of robust parameter levels into the HSA as initial population intervals, effectively restricting and refining the design space based on the effect of various design variables on the objective function. The optimal results for the shape optimization problem were obtained by applying the HSA. The findings of the shape optimization problem demonstrated that the proposed

approach was highly competitive and could be considered as a viable alternative for solving real-world optimization problems, delivering better solutions compared to other state-of-the-art optimization methods.

The issues with the previous studies that have attempted to combine HSA with local search algorithms is firstly that it lacks of a precise balance between the two approaches, leading to local search algorithms being solely responsible for optimization. Secondly, the previous research also failed to address the problem of adjusting HSA's parameters based on the behavioural neighbourhood movements of local search. Thus, this study considered both the modification and hybridization of HSA, therefore, in the modification phase, this study investigated the Markov Chain Theory to be implemented in the HSA operators that consider two dimensions of search space (iteration position and solution number), behavioral status objective function obtained from solution, linkage mechanism from previous fixed parameters of HSA and all parameters (iteration size, HMCR, RCR, and PAR). For the hybridization phase, this study explored GDA to be hybridized with HSA. Both phases were described in Sections 2.6 and 2.7 respectively.

2.6 Markov Chains Theory

Markov Chain theory is a widely recognized and respected method for analysing and comprehending the behaviour of systems that are subject to change over time. The theory proposes that the future state of a system is contingent solely on its current state and not on any previous states. The method is employed to segment a system into various states and subsequently determine the probability of transitioning between those states. This approach can be utilized to predict future behaviour and discern long-

term patterns across a broad range of systems. The theory is well-established in the fields of mathematics, statistics, and computer science, and is frequently applied in various disciplines such as finance, weather forecasting, and social networks (Karras et al., 2022). There are numerous scholarly articles (Seabrook & Wiskott, 2022; Ye et al., 2019) and books (Pandey, 2022; Tao et al., 2017; Yang et al., 2020) that offer a more in-depth understanding of the theory and its applications.

Markov Chain theory is closely related to another method called Markov Decision Process (MDP). MDP is a powerful theoretical frameworks for modeling decision-making problems under uncertainty. Both MDP and Markov Chain are based on the concept of Markov property, which the future state depends only on the current state and not on any previous states. However, while Markov Chain is mainly used to model and analyse the behaviour, MDP is used to model decision-making processes. MDP consists of a set of states, a set of actions, and a set of probabilities that describe the transitions between states (Aït-Sahalia & Hansen, 2010; Ovrutsky et al., 2014; Oweiss, 2010; Ren et al., 2017; Yang, 2020). The decision maker (also known as the agent) is assumed to have some control over the process, and can choose which actions to take at each state. The goal of the decision maker is to find the optimal policy, which is a strategy that maximizes some measure of performance over time. MDP and Model-based Reinforcement Learning (RL) are two closely related techniques for modelling decision-making processes. Both methods involve an agent that interacts with an environment and learns to make decisions that optimize a particular performance metric over time (Moerland et al., 2023).

In a Markov Chain model, the probability values are restricted to the range of 0 to 1. To conform to this range, the values must be normalized using the formula in Equation 2.17.

$$\text{Normalized Value} = \frac{Value - Minimum(n)}{Maximum(n) - Minimum(n)} \quad (2.17)$$

Where

$Value$ – the current state

n – set of values

$Minimum$ – the minimum value within set of values

$Maximum$ – the maximum value within set of values

An example of a set of number {34, 52, 23}, in which the minimum value is 23, maximum value is 52 and the current value to be normalized is 34. The normalized value using formula in Equation 2.17 will produce a value of 0.38 as shown below.

$$\text{Normalized Value} = \frac{34 - 23}{52 - 23} = \frac{11}{29} \approx 0.38$$

An example of a basic MDP can be illustrated by choosing two solutions that are represented as A and B, as depicted in Figure 2.11. To select between solutions of A and B, there are two options: either select A on the first move and then select B on the second move or select B on the first move and still choose B on the second move. The probability of these two options is calculated by multiplying the probabilities of each step and adding them together, which results in 0.3 x 0.7 + 0.7 x 0.2 = 0.35. Alternatively, the probability of selecting A after two moves is calculated by

multiplying the probabilities of staying at A or choosing from A to B on the first move, and then multiplying the probabilities of staying at A or choosing from B to A on the second move. This results in 0.3 x 0.3 + 0.7 x 0.8 = 0.65. Since there are only two solutions in the process, if the process is not on A, it must be on B. Therefore, the probability of selecting B after two moves is 1 - 0.65 = 0.35. It is important to note that these probabilities are determined using the stochastic rule, which is based on the probability of the current state, rather than the heuristic rule, which is based on previous experiences or observations.

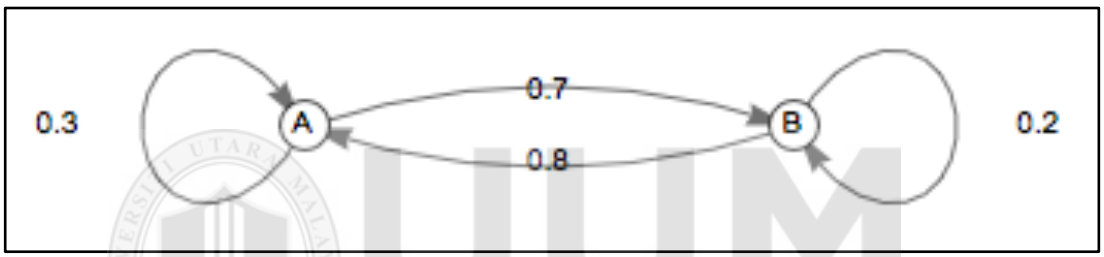


Figure 2.13. States A and B (Brilliant, 2020)

MDP has two benefits. First, it able to model decision outcomes as stochastic events and the fact that the probability of future states depends solely on the current state (Kamrani et al., 2020). Secondly, MDP is characterized by fully observable states and outcomes that are affected by decision makers (Rizk et al., 2018). For example, MDP can be used to classify states based on the presence and proximity of objects in order to avoid congestion in autonomous vehicles (Kamrani et al., 2020). In this case, the final decision state can be estimated based on the objectives (states) of the surrounding objects. Another example of the application of MDP is in the estimation of groundwater quality, where a MDP is used to calculate a weighted average of groundwater quality

grades (Su et al., 2019). In summary, the key property of MDP is that future decisions and outcomes are solely dependent on the current state of the process.

Based on literature reviews, it appears that the use of MDP for timetabling problems has not been widely explored. However, MDP have been successfully applied in the scheduling problem area. For example, Chen et al. (2013) used an MDP-based adaptive scheduling approach to solve the problem of storing scalable videos. Edrisi et al. (2019) proposed an adaptive dynamic route selection model using MDP while Şahin (2017) used MDP to analyse delay distributions in train schedules, and Patrick (2012) proposed the use of MDP to determine optimal outpatient scheduling. These studies demonstrate the potential of MDP to predict paths, policies, or the shortest path to improve scheduling problems.

The following are the advantages based on previous studies (Chen et al., 2013; Edrisi et al., 2019; Patrick, 2012; Şahin, 2017) of using MDP which could enhance the performance of the algorithms:

1. Modelling decision outcomes as stochastic events: MDP model the decision outcomes as probabilistic events, which allows for more accurate predictions of future states by taking into account the inherent randomness of the problem (Chen et al., 2013; Edrisi et al., 2019; Patrick, 2012; Şahin, 2017).
2. Incorporating the concept of temporal dynamics: MDP takes into account the temporal dynamics of the problem by modelling the probability of transitioning from one state to another over time, which is a crucial aspect of many decision-making problems (Chen et al., 2013; Edrisi et al., 2019; Patrick, 2012; Şahin, 2017).

3. Providing a framework for optimal decision-making: MDP provide a framework for making optimal decisions by considering the expected long-term rewards or costs associated with different actions. This allows for the identification of the most optimal policy for a given problem, which can greatly improve the performance of the algorithm (Chen et al., 2013; Edrisi et al., 2019; Patrick, 2012; Şahin, 2017).
4. Handling uncertainty: MDP can handle uncertainty by considering multiple possible states and outcomes, which makes it more robust to changes in the problem. This is especially useful in problems where the state of the system is not fully observable or where the model of the problem is uncertain (Chen et al., 2013; Edrisi et al., 2019; Patrick, 2012; Şahin, 2017).
5. Improving computational efficiency: MDP can improve the computational efficiency of algorithms by reducing the number of states or outcomes that need to be considered, which can be computationally expensive (Chen et al., 2013; Edrisi et al., 2019; Patrick, 2012; Şahin, 2017).
6. Incorporating prior knowledge: MDP can incorporate prior knowledge about the problem into the decision-making process, which can improve the accuracy of the algorithm. This can be done by specifying the initial state probability distribution or transition probabilities that reflect the prior knowledge (Chen et al., 2013; Edrisi et al., 2019; Patrick, 2012; Şahin, 2017).

MDP and Markov Chains are often thought of as mathematical systems, but they are also a powerful theoretical framework that can be applied to a wide range of fields. Based on the features stated above, MDP and Markov Chains are suitable tools for studying problems in many fields such as physics, economics, computer science, and

many others. In addition, MDP and Markov Chains have been used to model variety of real-world domain systems such as weather patterns, stock prices, and biological systems (Levin et al., 2008).

Figure 2.14 provides states about the deceleration of driver's behaviour (left: naïve approach, right: MDP). The naïve approach considers what happened in the previous period and predicts the same thing will happen again. In the right figure, in state one, 49% of driver's decision is acceleration (blue colour) (increase in the rate or speed), 23% is deceleration (orange colour) (reduction in speed or rate) and 28% maintaining constant speed (grey colour). However, if use naïve approach or driver's decision (left figure), then perhaps deceleration (orange colour) is unable to be constant (fluctuate) to prevent a crowdedness and crash. It can be concluded that although the MDP is able to find high quality of decision, it tends to be over-decision comparing with model based on one state or heuristic. As KHE algorithm consists of several layers in times assignment component that sorted based on one heuristic rule, the MDP have been selected to improve the sorting layer method on KHE algorithm.

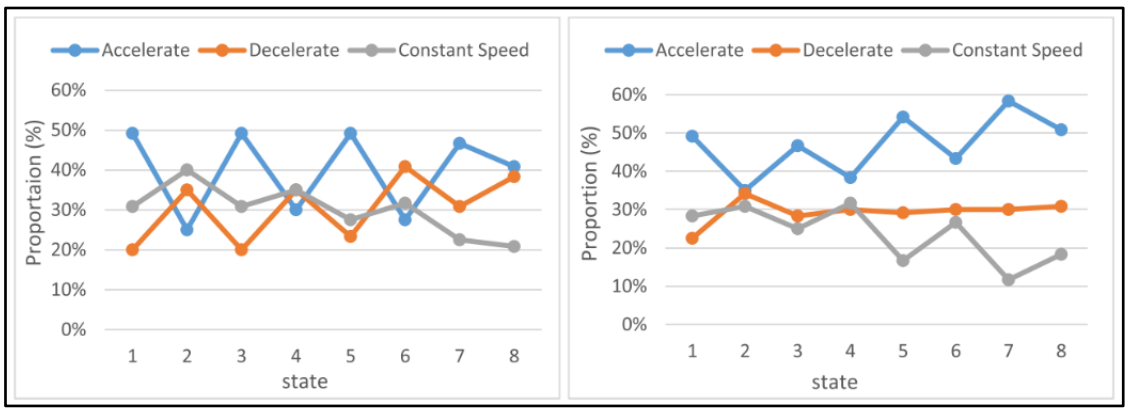


Figure 2.14. Example of the MDP Impact (left: model based one state, right: MDP) (Kamrani et al., 2020)

From this point of view, this study investigated the theory of MDP and Markov Chains to be implemented in the proposed construction and improvement phases. In the improvement phase, besides of using MDP, the GDA also applied. Next section explains GDA which hybridized with HSA.

2.7 Great Deluge Algorithm (GDA)

There are several local search methods that can be hybridized with HSA include GDA, SA, HC. However, GDA demonstrated the superiority over SA and HC based on two facts. Firstly, GDA can accept the worst solution in order to increase the likelihood of finding the best solution and avoid stagnation. Compared to HC that only accept best solution which does not offer alternative paths that are worse than the current solution (Du & Swamy, 2016). Secondly, GDA uses the water level rate (linear and nonlinear), which is linked to the objective function, compared to SA that uses a probability which may negatively impact the performance of the algorithm and prevent it from finding a good solution (Eng et al., 2020; Junn et al., 2017).

GDA is proposed by Dueck (Dueck, 1993). It is a single solution-based metaheuristic for continuous global optimization which accept inferior solutions depend on an acceptance rule through the solution search method. This helps to escape from local optima (Du & Swamy, 2016).

The analogy presented in Figure 2.15 is commonly used in optimization problems and can be described as follows (Baykasoglu, 2012). Imagine a human (*S*), representing the current solution, hiking up a hill, which is depicted as a maximization problem. The objective is to reach the highest point on the hill. However, the hiker wants to avoid

getting their feet wet, so they cannot cross any streams or wetlands. As the hiker progresses, they may become stuck in a local optimum, where the slope of the hill is not steep enough to reach the global maximum, but any other direction would lead to wet feet (S^d). To overcome this, the hiker must use information about the rises in water level, represented by $S^a, S^b, or\ S^c$, to discover a path that will allow them to escape from the local optimum and continue towards the global maximum. This analogy is often used in the context of optimization algorithms, where the objective is to find the best solution to a problem within a large search space. Similar to the hiker in the analogy, the GDA can explore the search space and avoid getting trapped in local optima.

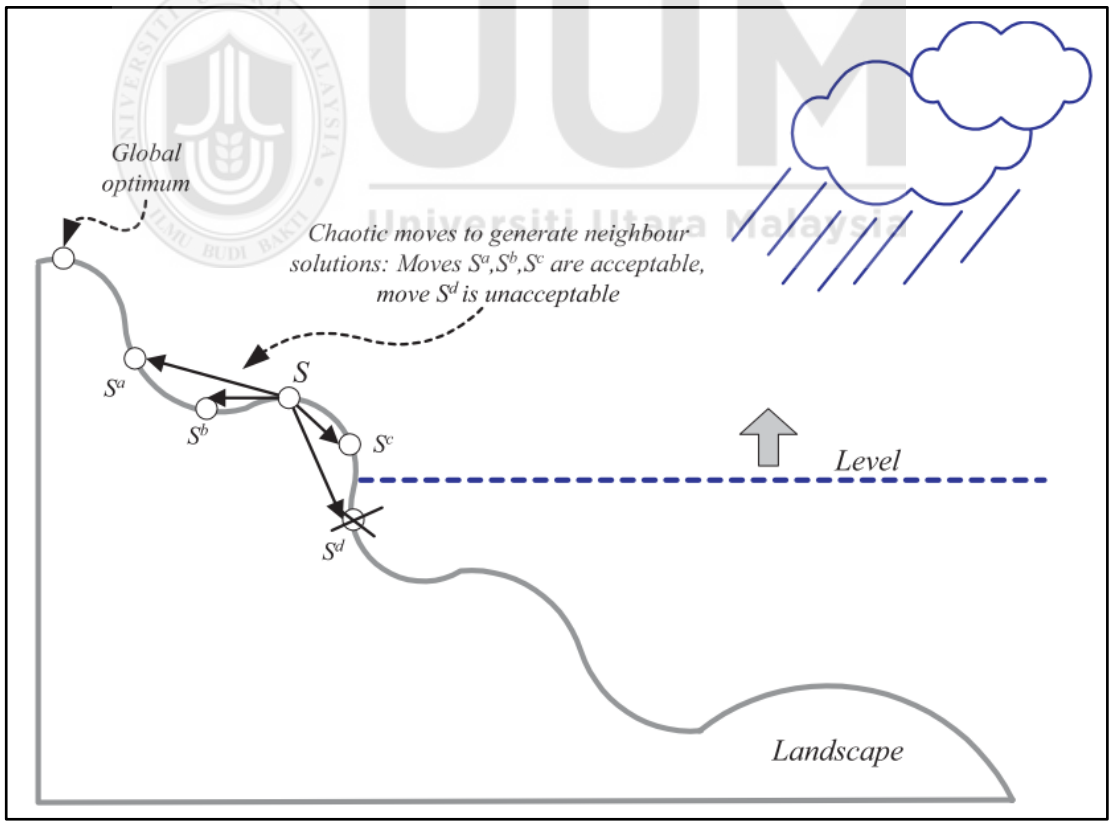


Figure 2.15. Illustrative representation of GDA search (Baykasoglu, 2012)

Figure 2.16 describes the process of GDA that starts with an initial solution quality as represented in Step 1. At Step 2, the value of the computation speed ($rainSpeed$) and total iteration ($totalIter$) are set. The initial values of the decrease rate ($decayRate$), water level (B) and cost of initial solution (objective function) are calculated in Step 3. The water level (B), which is often selected to be in line with the objective function, serves as the starting point for the algorithm. As the algorithm progresses, the water level (B) is linearly dropped by $decayRate$ throughout the search, and the current solution's objective function is adjusted as the water level (B) is reduced (or raised) until the convergence is reached. At each iteration, a new solution is created using a neighbourhood structure (NS) as shown in Step 4. An enhancing solution, $f(S_{new})$ is accepted all the time, which is better than existing solution, $f(S)$ ($f(S_{new}) \leq f(S)$), while decreasing one is taken if it is better than the original water level ($f(S_{new}) \leq B$). The water level (B) is reduced by the formula in Equation 2.18. The iteration will end when the $totalIter$ equals to zero.

$$B = B - decayRate$$
$$decayRate = f(S) \times \frac{rainSpeed}{totalIter} \quad (2.18)$$

Where,

$f(S)$ - the cost of the existing solution

$decayRate$ - specified decay rate

B - the lowest estimated penalty related to the best solution

$totalIter$ - the value of iterations

$rainSpeed$ - responsible for the computation speed

Overall, GDA will iteratively adjusts the water level to explore the search space and find a better solution to a given problem. By reducing the water level at a specific rate and exploring the neighbourhood of the current solution, the GDA aims to avoid getting trapped in local optima and converge to a global optimum.

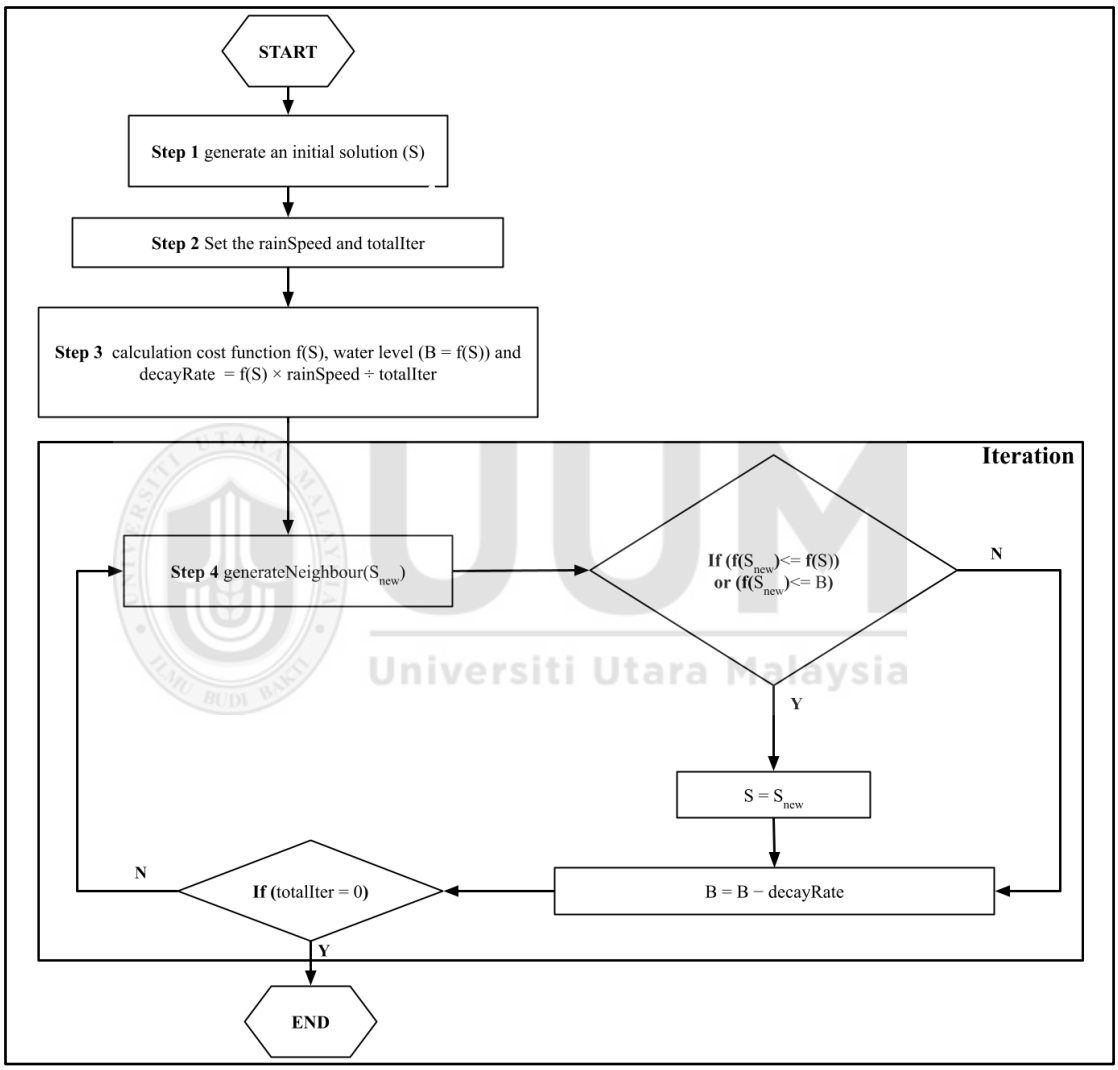


Figure 2.16. Great Deluge Algorithm Flow Chart Adapted from Eng et al. (2020)

There exist some studies focused on improving the fundamental of GDA method that used linear decay rate. The modification made by Landa-Silva & Obit, (2008b), is specifically intended for solving course timetabling problems using the non-linear GDA

method. The non-linear decay rate for the water level, with its various parameters δ, min, max and β, are used to optimize the solution by controlling the rate at which water level (B) is added to the search space as shown in Equation 2.10. The parameters "min" and "max" play a crucial role in controlling the decay rate's speed and the search process's pace, specifically through the expression $exp^{-\delta(rnd[min,max])}$. Higher values of min and max lead to faster water level reduction, resulting in faster search progress but potentially getting trapped in local optima early on. Experimentation was used to determine the parameter values in Equation 2.19, with δ that assigned as small, medium, and large instances. The value of β was set to 0 for all problem instances. The values of min and max were chosen based on the problem instance's size, with min = 10000 and max = 20000 for small problems and min = 100000 and max = 300000 for medium and large problems.

$$B = B \times \left(exp^{-\delta(rnd[min,max])}\right) + \beta \tag{2.19}$$

Mafarja & Abdullah, (2011) divided the GDA search space into three equal-sized regions or intervals, which was used to solve attribute reduction problems. Each interval is assigned a level value, which is referred to as water level1, level2, and level3. The levels are updated using three different increasing rates, β. Additionally, three β values ($\beta 1$, $\beta 2$, and $\beta 3$) are introduced to update the levels at each of the three regions. The effectiveness of modified GDA was evaluated on standard benchmark datasets, and the experimental results demonstrate that their approach is capable of generating high-quality results that were comparable to those produced by other well-known optimization algorithms such as TS, ACO, GA, SA, scatter search, basic GDA,

composite neighbourhood structure, hybrid variable neighbourhood search algorithm, and a constructive hyper-heuristics.

Mafarja & Abdullah, (2014) considered the Fuzzy Modified GDA (Fuzzy-mGD) as a Multi Input Single Output (MISO) dynamical system for solving attribute reduction problems, by sampling the Fuzzy-mGD outputs and acting on its inputs according to the fuzzy rules. The fuzzy controller was used to update the level by applying different values through the search process instead of using one increasing rate (as in the original GDA) or three increasing rates. The GDA search space was divided into three equal areas, each representing a fuzzy set (low, medium, and high) in the fuzzy logic system. The controller took two inputs, the trial solution and the best solution, which were connected to the general terms: low, medium, and high (corresponding to fuzzy set meanings). A set of rules that linked the input variables (trial and best solution) with the single output variable (β) was built according to the fuzzy rules. The experimental results demonstrated that their approach generated high-quality solutions for the well-known benchmark datasets from the attribute reduction literature, which were comparable to those produced by other popular optimization algorithms, such as TS, ACO, GA, SA, scatter search, basic GDA, composite neighbourhood structure, hybrid variable neighbourhood search algorithm, and a constructive hyper-heuristics.

In the work of Landa-Silva & Obit, (2008b), GDA's nonlinear decay rate is isolated from the feasibility of the objective function. For instance, if the objective function requires two feasibilities (hard and soft constraints), then both nonlinear GDA cannot perform a good decay water level rate based on these two feasibilities. In such cases, the decay water level rate should be divided based on the feasibility of the objective

function. In addition, nonlinear GDA has the added complexity of minimum and maximum parameter settings based on the dataset size. In the work of Mafarja & Abdullah (2011), the dividing of search space into equal-sized regions or intervals can be problematic in some cases because it may not capture the underlying structure of the search space. For example, if the search space has regions that are more important or more likely to contain optimal solutions, dividing it equally may not allow the algorithm to explore those regions effectively. In the work of Mafarja & Abdullah (2014), the used of fuzzy logic in a modified GDA can increase the complexity of the algorithm, which can lead to longer execution times and increased memory usage. This can make it more difficult to use the algorithm in real-world applications where efficiency is important. Additionally, incorporating fuzzy logic may require a higher level of expertise in both of GDA and fuzzy logic. Another challenge with using fuzzy logic in optimization algorithms is that the optimal settings for the fuzzy logic parameters may difficult to be determined. These parameters can greatly affect the performance of the algorithm, so it is important to select them carefully. There may not be a clear way to determine the best values for these parameters. Finally, fuzzy logic may not be appropriate for all problems or situations so as to address the limitations of optimization algorithms.

This study aims to address the deficiencies of prior works and enhance the GDA to optimize the objective function, which includes both hard and soft constraints. Specifically, the study seeks to maintain a linear correlation between the decrease in water level decay rate and the objective function, while simultaneously minimizing algorithmic complexity.

2.8 Rationale for the Selected Methodology

The methodology for this research focuses on hybridizing HSA with GDA to solve HSTP using XHSTT data instances. This approach addresses specific inefficiencies in the timetabling process, particularly those related to time assignment, static parameter limitations in HSA, and the need for enhanced search space exploration. Each aspect of the methodology is designed to meet the research objectives, providing a robust solution to the challenges of HSTP.

A key element of the methodology is the introduction of a model for heuristic rules in the time assignment stage, which is essential for overcoming inefficiencies in the KHE construction algorithm. KHE, widely used for solving HSTP, especially with XHSTT datasets, often struggles with inefficient layer arrangement during scheduling. By introducing a model for heuristic rules, the process of assigning events, such as teacher and class schedules, to time slots becomes more efficient. Heuristic approaches are particularly well-suited to handling complex scheduling problems like HSTP, as they provide flexibility in dealing with complex constraints such as teacher availability, room capacity, and student preferences. This enhancement optimizes the time assignment stage, ensuring that resources are better allocated while maintaining adherence to both hard and soft constraints. The introduction of these heuristic rules not only improves the speed of the scheduling process but also enhances the overall performance of the KHE algorithm, making it more responsive to the specific demands of high school timetabling.

Another essential aspect of the methodology focuses on addressing the limitations of static parameter settings in HSA, which often lead to suboptimal performance. Static

parameters reduce the algorithm's ability to explore the solution space effectively, especially in complex problems like HSTP. To counter this, an adaptive mechanism recalculates parameters, based on factors such as iteration position, solution number, linkage parameters, and the behavioral status of the algorithm. This adaptive recalibration allows HSA to adjust its balance between exploration and exploitation throughout the optimization process. Such adaptability is critical in solving HSTP, where problem instances vary significantly. Different data instances from the XHSTT dataset may require varying search intensities, and the adaptive mechanism ensures that the algorithm can flexibly adjust to these differences. This adaptive parameter setting improves the algorithm's efficiency in navigating the search space, ensuring that better solutions are found without the need for extensive manual tuning of parameters.

The hybridization of HSA with GDA addresses the challenge of limited search space exploration and insufficient exploitation of potential solutions. While HSA is effective at exploring a broad solution space through its vigorous adaptability, it may struggle to refine these solutions optimally without additional support. This is where GDA complements HSA. The GDA algorithm focuses on gradually improving solutions over time by tightening the acceptance criteria for new solutions, ensuring that the search process avoids premature convergence to local optima. The introduction of two decay rates in GDA allows for a gradual shift from exploration to exploitation, ensuring that the algorithm does not settle on suboptimal solutions early in the process. This hybrid approach ensures that both the broad exploration provided by HSA and the focused refinement offered by GDA work in tandem, creating a more robust search process for solving the HSTP.

This hybridization leverages the complementary strengths of HSA and GDA. HSA's adaptability and broad search capabilities are well-suited to navigating the complex, multimodal solution spaces of HSTP, while GDA provides the necessary refinement to ensure that high-quality solutions are found. The use of two decay rates further enhances the balance between exploration and exploitation, allowing the algorithm to shift focus gradually and methodically as the search progresses. This approach is particularly effective for handling the XHSTT dataset, which includes real-world constraints such as teacher availability, room capacities, and student preferences. By balancing these constraints while optimizing both hard and soft objectives, the hybrid algorithm ensures that solutions are both feasible and of high quality.

In conclusion, the chosen methodology provides a comprehensive solution to the HSTP. By introducing a model for heuristic rules in the time assignment stage, overcoming static parameter inefficiencies with adaptive settings, and enhancing search space exploration through the hybridization of HSA and GDA, this approach offers a powerful framework for solving complex, real-world timetabling problems. The integration of these elements not only meets the research objectives but also contributes a solution to the broader field of timetabling and optimization challenges.

2.9 Summary

This chapter comprehensively outlines the high school timetabling problem and its associated datasets. The datasets consist of Greek, Brazilian, Lectio, OR-library and. XHSTP high school datasets. The chapter also provides an overview of the most commonly used construction and optimization methodologies for solving high school timetabling. Furthermore, a comparison of the best results of optimization

methodologies obtained in previous literature is presented. Next, the review of HSA, Markov Chain theory and GDA with their applications and modifications in previous studies were highlighted. Finally, the rationale of choosing the proposed methodology were also described. Next chapter describes the methodologies used to address the objectives of the thesis.

CHAPTER THREE
RESEARCH METHODOLOGY

3.1 Introduction

This chapter elaborates the methods used for achieving the research objectives. Section 3.2 presents the overview of the research framework that consists of three phases with its input, activities and output. The initial solution construction and its enhancement are described in Section 3.3 while improvement phase (Section 3.4) that consists of five subsections i.e., basic HSA, enhancement of HSA, basic GDA, modified GDA and hybridization of enhanced HSA with modified GDA are described in Sections 3.4.1, 3.4.2, 3.4.3, 3.4.4 and 3.4.5 respectively. Next, Section 3.5 highlights the experimental and evaluation results. Finally, this Chapter is summarized in Section 3.6.

3.2 Research Framework

The research framework, as shown in Figure 3.1, is composed of three phases. The first phase is the construction in which the initial school timetable is created using dataset. In this phase, firstly the original KHE applied to produce the population of initial solution, followed by the enhancement of the KHE algorithm where the Markov Chain theory used to model the heuristic rules in order to improve the KHE algorithm. The result of this phase is an improved construction algorithm that will assist to improve the population of initial solutions. In the second phase, the basic of HSA implemented to improve the population of initial solution. Next, the basic HSA enhanced in terms of its parameters to become more adaptive. In addition, the GDA also applied to improve the initial solution in order to further modify the GDA. The modified GDA (MGDA) used to hybridize with enhanced HSA. The result of this phase is an enhanced of HSA that

hybridized with MGDA that will produce better solutions. Next, real-world datasets are produced. The actual school timetable is collected from a specific school, then analyzed and converted into the XML format of the XHSTT dataset. It is then solved using the construction and improvement algorithms described in the first and second phases.

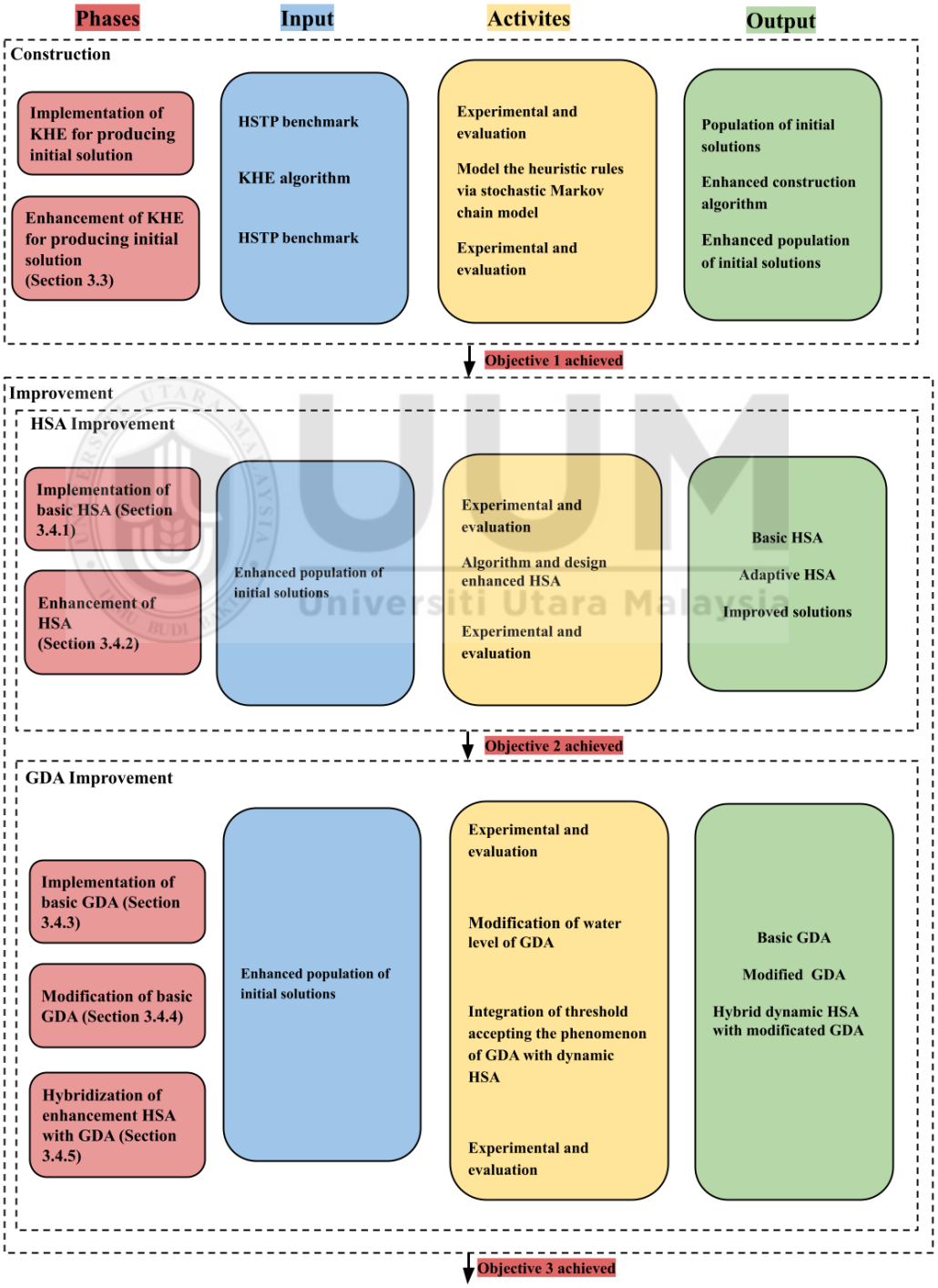


Figure 3.1. The Research Framework

3.3 Initial Solution Construction

This phase consists of two stages which starts with implementation of KHE algorithm for constructing the population of initial solutions. Next, the KHE algorithm enhanced in its phase to produce better results of the initial solutions population.

3.3.1 KHE Implementation

KHE involves four phases i.e., structure, time assignment, resource assignment and clean-up phases to generate a timetable (a solution), as described earlier in Section 2.4.1. In this construction phase, the HSTP solution (timetable) produced with no or minimum hard constraint violations but may usually have many soft constraint violations. This type of timetable solution is called the initial solution, which will later be improved in terms of reducing both hard and soft constraint violations in the improvement phase.

The KHE algorithm can be obtained from the KHE homepage[6], where installation and usage instructions are provided. The website also provides information on the data types supported by KHE, as well as common operations applicable to these types. KHE is a publicly available ANSI C library and designed to offer a robust and efficient foundation for solving high school timetabling problems in XHSTT format. KHE users could manipulate XML files, generate solutions, and modify time and resource assignments using any preferred algorithms. The current solution cost is continually maintained and accessible through a constraint propagation network that is hand coded. While intended for practical applications, KHE also serves as a research tool, and future

[6] Home Page of KHE A Software Library for High School Timetabling http://jeffreykingston.id.au/khe/

versions are not constrained by compatibility requirements with previous versions. At the end of KHE implementation, a population of initial solutions were produced. The KHE algorithm was run with 39 XHSTT datasets as explained earlier in Section 2.3.2.5. The experimentation and evaluation of the KHE implementation explained in Section 3.6.

3.3.2 KHE Enhancement

In the experimentation of original KHE algorithm, the process of sorting layers for the time assignment phase relied on a single heuristic state (whether based on duration, demand or index), which contributes to increase the minimum-cost matching of meets to times in sorting layers. From this point of view, the KHE algorithm was enhanced in the sorting layers of time assignment phase by employing the Markov Chain theory. Markov Chain theory can be applied to model a probabilistic approach and combine several heuristic states together to come out with one probabilistic value that may give different decisions to guide the search space or sorting methods.

Figure 3.2 shows the original KHE of the time assignment phase sorting layers with single heuristic rule. If a layer has already been assigned timeslots (visited), then the unallocated layer should be prioritized for sorting. Otherwise, the layers sorting will proceed to the red zone section. The red zone shows how the layer 1 and layer 2 are sorted by comparing using single heuristic (duration, demand or index). If the duration value of layer 1 is equal to duration value of layer 2, the demand value of both layers compared. If the demand value of layer 1 is equal to the demand value of layer 2, the index value of both layers compared. If the index value of layer 2 is higher than index value of layer 1, the layer 2 will have high priority to be assigned compared to layer 1.

If the demand value of layer 1 is higher than demand value of layer 2, the layer 1 will have high priority to be assigned compared to layer 2. If the duration value of layer 1 is higher than duration value of layer 2, the layer 1 will have high priority to be assigned compared to layer 2.

The enhancement of KHE for the time assignment phase sorting layers consists of two conditions: 1) when a layer has been allocated timeslots (visited), the sorting should give priority to the unassigned layer; and 2) when both layers have not been allocated timeslots, the priority of layers calculated based on combination of duration, demand and index values using Markov Chain theory. The layer with the greater value should be sorted first.

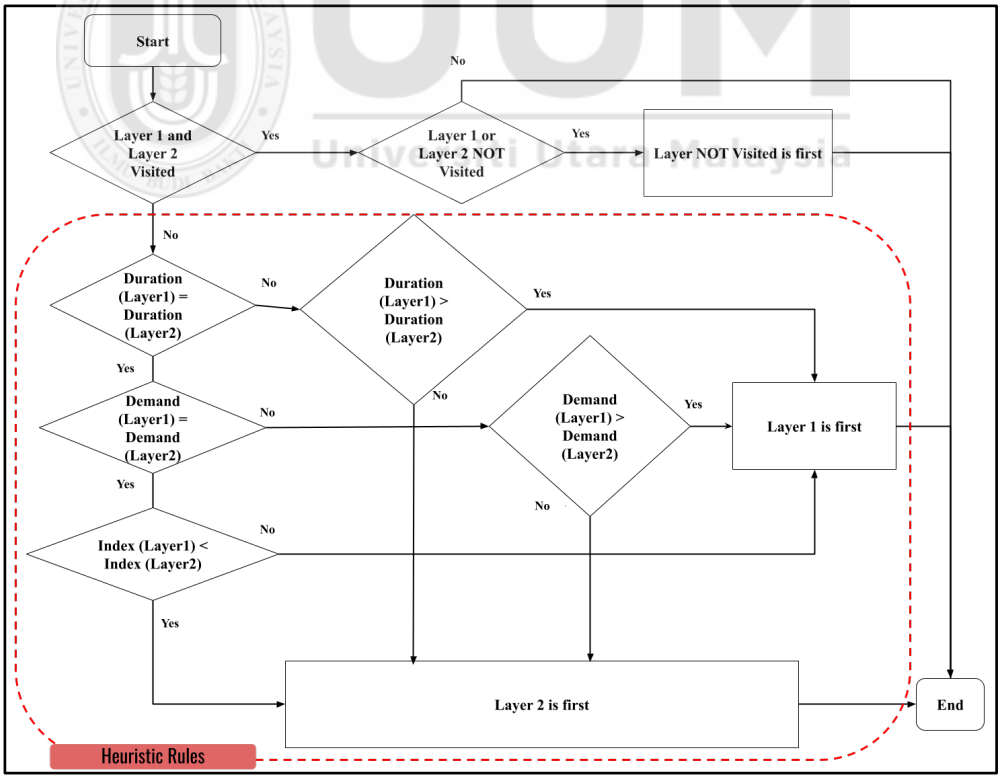


Figure 3.2. Flowchart before enhancing the sorting layers within the KHE time assignment phase

At the end of KHE enhancement, an improved population of initial solutions produced to facilitate the implementation of population-based metaheuristics, which is the first objective of this study. Same as in the implementation of original KHE, the KHE enhancement run and evaluated with 39 XHSTT datasets as described in Section 3.6.

3.4 Improvement Phase

This section describes the improvement of initial solutions population produced by the KHE algorithm in the enhancement stage. Firstly, the Preliminary Evaluation and Selection of Neighbourhood Structures for HSTP is described in Section 3.4.1. Next, the basic HSA is implemented as explained in Section 3.4.2, followed by the enhancement of HSA in Section 3.4.3. Section 3.4.4 outlines the basic GDA implementation, while Section 3.4.5 explains the GDA modification. Finally, the hybridization of the enhanced HSA with MGDA is highlighted in Section 3.4.6.

3.4.1 Preliminary Evaluation and Selection of Neighbourhood Structures for HSTP

The Neighbourhood Structures (NS), detailed in Section 2.4.3, formed the foundation of the preliminary experimentation phase aimed to evaluate 12 NS for solving the HSTP, using the BrazilInstance5 dataset as shown in Table 3.1 and Figure 3.3. This phase involved applying each NS under identical conditions across multiple datasets to assess their solution quality, measured using a score where lower values indicated better results. Significant variations in performance were observed. For instance, structures like MeetsSwap (0.00083) and BlockMeetsSwap (0.00099) achieved the best outcomes, while others, such as TaskEjectingMoveResource (0.00136), performed moderately but were retained due to their relative strength. On the other hand, poorly

performing NS, such as ResourcePairReassign and NodeMeetSwap, were excluded from further analysis as their higher scores indicated suboptimal solutions.

Table 3.1

Preliminary Experimentation Results Table

NS	Solution Quality	Performance	Selected for Further Analysis?
MeetsSwap	High	Effective	Yes
TasksSwap	High	Effective	Yes
MeetsMoveTime	High	Effective	Yes
TasksMoveResource	Moderate	Effective	Yes
BlockMeetsSwap	High	Effective	Yes
KempeMeetMove	High	Effective	Yes
KempeMeetMoveTime	High	Effective	Yes
TaskEjectingMoveResource	Moderate	Effective	Yes
EjectingMeetMoveTime	Moderate	Effective	Yes
EjectingMeetMove	High	Effective	Yes
ResourcePairReassign	Low	Suboptimal Solutions	No
NodeMeetSwap	Low	Suboptimal Solutions	No

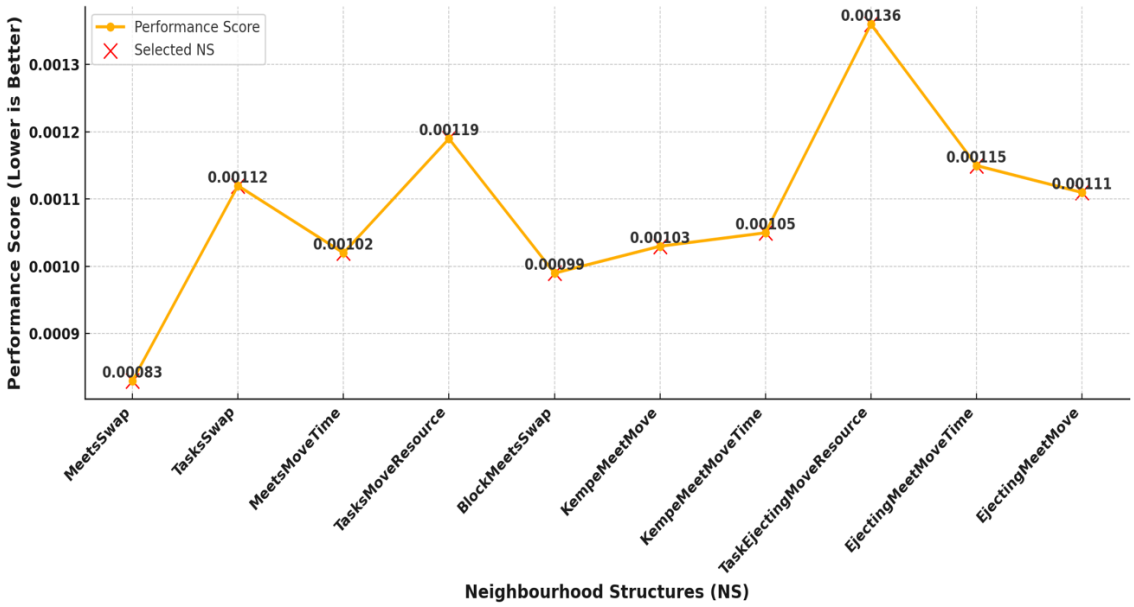


Figure 3.3 Performance scores of NS on BrazilInstance5 dataset

This evaluation process revealed critical insights into the behavior of each NS in improving solutions, particularly in minimizing conflicts and enhancing schedules. Based on these findings, only the 10 NS that demonstrated consistent potential for producing better-quality solutions were selected for further experimentation. The selected NS—MeetsSwap, TasksSwap, MeetsMoveTime, TasksMoveResource, BlockMeetsSwap, KempeMeetMoveTime, KempeMeetMove, TaskEjectingMoveResource, EjectingMeetMoveTime, and EjectingMeetMove—were highlighted in red in the graph, as they represented the most promising options for achieving improved efficiency and solution quality in subsequent stages of experimentation.

This preliminary experimentation phase is applied to all phases, including the basic HSA, enhancement of HSA, basic GDA, GDA modification, and hybridization, to ensure consistency in evaluating their effectiveness.

3.4.2 Basic HSA Implementation

In this section, the basic HSA algorithm as explained in section 2.5 implemented. Several parameters are set, including NS, HMCR, RCR, PAR, MI, and HMS. The best values for HMCR (0.80), RCR, PAR and MI (500) are set based on the number of NS and preliminary experimentation above. RCR and PAR are divided into several sub-parameters based on number of NS such as RCR1 = 0.75, RCR2 = 0.50, RCR3 = 0.25, PAR1 = 0.75, PAR2 = 0.60, PAR3 = 0.45, PAR4 = 0.30 and PAR5 = 0.15. The basic HSA algorithm was run with 39 XHSTT datasets. The description of the XHSTT datasets is presented in Section 2.3.2.5. The experimentation and evaluation of the basic HSA implementation explained in Section 3.6, where the performance of the algorithm

analyzed in terms of solution quality and computational efficiency. The details of basic HSA implementation are further explained in Chapter Five.

3.4.3 Enhancement of HSA

In the experimentation of original HSA algorithm, the basic HSA algorithm separated the search space for the parameters (HMCR, PAR, and RCR) from the problem's objective function. This caused a reduction in the efficacy of the parameters within the search space, which in turn increased the likelihood of the solution getting stuck in the search space and not progressing further. Thus, addressing these problems involves several strategies. Firstly, integrating the iteration position with the current solution number (which represents two dimensions of the search space) during the HSA search process might yield better solutions. Secondly, the behavioral status of neighborhood moves (improve, equal improve, and not improve) from previous iterations can be utilized in subsequent iterations to track potential improvements in solution quality. Thirdly, the fixed parameters of HSA from previous studies can be adopted to generate adaptive parameters, guiding the parameter rates through the search space more effectively. Finally, all parameters of HSA (HMCR, PAR, and RCR) can be adaptively adjusted, allowing for the adaptive selection of neighborhood moves within each parameter, with the aim of reducing the overall search process time. From this point of view, the basic HSA algorithm enhanced in its parameters by employing the combination between iteration position and solution number, behavioural status of neighbourhood moves, and parameters linkages. Based on these three matters, four parameters (HMCR, PAR, RCR and MI) adaptively calculated during the search space. However, the population size (HMS) will not be adaptively changed as it was set by the

construction algorithms (KHE). From this point, the enhancement of HAS is named as adaptive HSA.

For adaptive HSA, each solution is characterized by its status, which can be "improve", "equal-improve" or "non-improve". Initially, all solutions are set to "improve" status. The status of each solution is updated after each iteration of the HSA, based on the outcome of applying NS to the solutions. The HMCR, RCR, PAR and MI calculations are further explained in Chapter Five. The adaptive HSA algorithm was run with 39 XHSTT datasets. The experimentation and evaluation of the adaptive HSA explained in Section 3.6, where the performance of the algorithm analyzed in terms of solution quality and computational efficiency. This section addresses the second objective of this study.

3.4.4 Basic GDA Implementation

This section describes the implementation of the basic GDA algorithm, which was previously explained in Section 2.7. The reason for the basic GDA implementation is to investigate the nature of its in solving HSTP. Initially, the GDA algorithm selects the best (lowest) objective function from the initial population of solutions which assign it to the water level. Next, the rain speed is set to 0.4 (set based on the preliminary experimentation). The decay rate is calculated according to Equation 2.18. New solution is generated iteratively and accepted if the cost of the solution is less or equal to the water level. The water level gradually decreased over iteration based on decay rate. The algorithm terminates after reaching a maximum number of iterations. Section 3.6 details the experimentation and evaluation of the basic GDA algorithm, which was

run using 39 XHSTT datasets. Chapter Six will provide further details on the implementation of the basic GDA algorithm.

3.4.5 Modified GDA

During the experimentation of the original GDA, it was found that the basic GDA had separated the water level and decay rate of search space from the objective function of the problem. Regarding previous research, GDA has insufficiently addressed the non-linear connection between water level acceptance and the objective function. As a result, there was a disparity in the synchronization between the water level, decay rate, and search space of the problem, which increased the likelihood of the solution being stuck in the search space and not making further progress. To address this issue, the basic GDA improved by incorporating objective function of the problem, water level and decay rate into its parameters by dividing the objective function of the problem based on soft and hard constraints.

In the initial stages of the algorithm, the decay rates are calculated by multiplying the current violation of soft or hard constraints by the total number of iterations and dividing the result by the rain speed. During each iteration of the algorithm, new values for the water level of the soft and hard constraints are generated and gradually decreased using their respective decay rates. To measure this, the proposed work is compared with previous research by evaluating how the new solution path and water levels decrease linearly together. Thus, the analysis investigates the impact of water levels on the previous adaptive GDA solution search by focusing on the decay rates across sizes of iterations (50, 100 and 200). It examines the relationship between water level acceptance and solution cost (objective function) to understand how this linear

connection enhances the likelihood of finding multiple solution paths, thereby promoting greater exploration. Section 3.6 provides a detailed of the experimentation and evaluation of the MGDA using 39 XHSTT datasets. Further details on the implementation of the MGDA are presented in Chapter Six.

3.4.6 Hybridization of Adaptive HSA with Modified GDA

While testing the MGDA, it was discovered that the MGDA exhibited insufficient diversification of population solutions and excessive intensification of neighborhood movements. This resulted in an observable difference in behavioral status between the water level and NS, thereby increasing the likelihood of the solution becoming trapped in the local search space. To address the aforementioned issue, the water level of the MGDA integrated with the adaptive HSA parameters. The detailed explanation of the hybridization process between the adaptive HSA and MGDA is presented in Chapter Six. In Section 3.6, a comprehensive analysis of the experimentation and evaluation of the hybridization process is provided, based on 39 XHSTT datasets. This part of the study focuses on achieving the third objective.

3.5 Experimental Evaluation

This section explains the computational environment and results evaluation of initial solution construction and improvement phases.

The computational environment for the above processes (Sections 3.3 and 3.4) performed with the following aspects:

1) The development processes carried out using C language that coded in Atom editor.

2) The specification computer used is AMD Ryzen 7 3700X 8-Core with 3.89 GHz processor and 16 GB RAM.
3) The Windows subsystem for Linux is used to run the experiments based on Ubuntu 20.04 terminal environment.
4) All the improvement phases (presented in Sections 3.4.1, 3.4.2, 3.4.3, 3.4.4 and 3.4.5) were executed together with enhanced KHE (construction phase), respectively. This is due to produce better results for each improvement phase with different initial solutions and random seeds.
5) This study calculates the level gap to see the hard and soft constraints differences between the original KHE, enhanced KHE (Section 3.3.2) and Enhancement of HSA (Section 3.4.2) by using Equation 3.2, where it has three variables to present the value of the decreased level gap for soft and hard constraints. The lowest or equal to zero of level gap value shows that the current cost is better than the initial cost.

$$\downarrow GAP = \frac{(Current_{cost} - Lower_{bound})}{(Initial_{cost} - Lower_{bound})} \quad (3.2)$$

Where,

$Current_{cost}$ - the cost of soft and hard constraints produced by the enhanced KHE or Enhancement of HSA

$Lower_{bound}$ - the lower bound

$Initial_{cost}$ - the cost of original KHE

6) Rank Average: To compare the performance of different algorithms, their objective functions from their best runs on each instance in order, are used as a metric. The average rank is then computed by taking the mean of these rankings

as shown in Equation 3.2. Value of rank is set based on number from the best (1) to the worst (n) of objective function. The approach with the lowest rank average is considered the best performing method.

$$\text{Average rank} = \frac{\sum_i^n \text{rank}_i}{m} \tag{3.3}$$

Where,

m-is number of datasets

n-is number of approaches

i- the current approach

7) For Section 3.4.2 (Enhancement of HSA), the experiment was performed/run five times with varying initial solutions and random seeds. The average of each hard and soft constraints violation calculated and compared to results (averages) of the-state-of-the-arts methods.

8) Producing the Real-world School Timetable Datasets: this part will present the conversion of real-world school timetable to the real-world datasets of HSTP based on the XHSTT dataset format. The school timetable that obtained from a primary school located in the north of Kedah, Malaysia, extracted its information to produce five elements which are *metadata*, *times*, *resources*, *events*, and *constraints*. This will make the school timetable as XHSTT dataset which named as MalaysiaSFBT that represent the acronym school name. Afterwards, the MalaysiaSFBT classified into three levels of difficulty which are easy (MalaysiaSFBT01), normal (MalaysiaSFBT02) and hard (MalaysiaSFBT03) based on the difficulty level assigned by the teachers. Next,

all the three datasets solved using the initial solution construction and improvement phases stated in aforementioned sections. These three solutions of datasets uploaded to HSEval[7] (High School Timetable Evaluator) for evaluation. The evaluator works by accepting solutions of each XHSTT dataset file in XML format. If the file is not in the correct solution format, the evaluator will return with error information. The experimental process was conducted using the same computational environment as discussed above from 1 to 4. A more comprehensive elaboration of this part presented in Chapter Seven.

The experimental cost evaluation, or objective function in this study evaluated and analyzed based on a single integral value of the hard and soft constraint violations. For instance, a value of *4.03447* denotes that the total number of hard constraint violations is 4, and the total penalty value of soft constraint violations is 3447.

The results of the original and enhanced KHE compared to each other. In addition, both results compared with the lower bound (an optimal solution or the best possible solution that can be achieved). The enhanced KHE compared to the state-of-the-art methods which consist of Mixed-Integer linear Programming (Kristiansen et al., 2015b), Matheuristics (Fonseca et al., 2016c), MaxSAT-based large neighbourhood search (Demirović & Musliu, 2017), A Hidden Markov Model Approach to the Problem of Heuristic Selection (Kheiri & Keedwell, 2017), and variable neighbourhood search (Fonseca et al., 2016c). The methods were selected for comparison because each of them were the best methods in their categories. The analysis of enhanced KHE results

[7] The HSEval High School Timetable Evaluator http://jeffreykingston.id.au/cgi-bin/hseval.cgi

consisted of two parts. The initial step involves calculating the level gap value, which ranges from zero to two, for both hard and soft constraints. This is done by taking the difference between the lower bound and the enhanced KHE result and dividing it by the difference between the lower bound and the initial solution result. The second step involves analyzing the Markov Chain equations in the enhanced KHE using two cases to demonstrate the limitations of the original KHE algorithm.

The process of adaptive HSA evaluation and analysis consists of two main steps: comparison and analysis, as shown in Figure 3.4. The comparison aims to see the quality of adaptive HSA results compared with enhanced KHE, basic HSA, current best-known results of metaheuristic (as listed in Table 2.12, such as VNS, SGVNS, SA+ILS, HSA-SA, SA-LAHC, SA-sf-LAHC, and RVNS, and GVNS), and state-of-the-art results (as listed in Table 2.13, such as IP, MIP, fix-and-optimize, and two-stage decomposition). The analysis part involves two aspects, which are level gap and average solution. To accomplish the level gap, the lower bound value is subtracted from the adaptive HSA outcome, and the resulting difference is divided by the difference between the lower bound and the initial solution result. In addition, the second strategy is used to evaluate the results based on average solution which run 5 times.

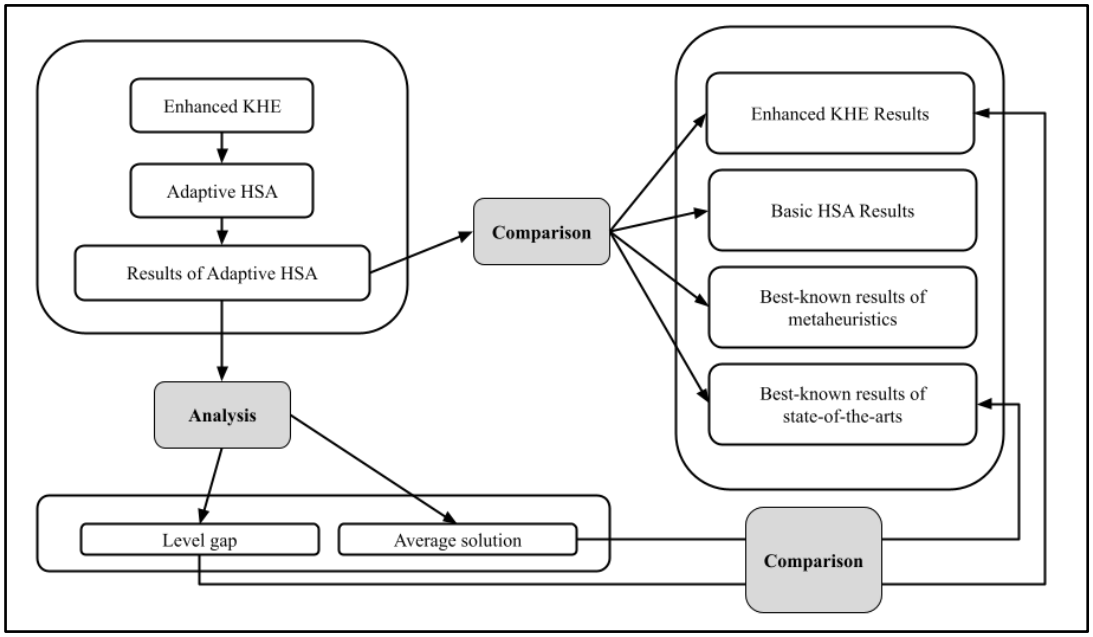


Figure 3.4. Evaluation and analysis of adaptive HSA

Figure 3.5 explains the evaluation and analysis process for both MGDA and hybridization of adaptive HSA with MGDA. The evaluation and analysis process for MGDA involves two main steps, namely comparison and analysis. During comparison, the objective is to evaluate the quality of MGDA results against enhanced KHE, basic GDA, the current best-known results of metaheuristic, and state-of-the-art results. The analysis part involves examining how soft and hard water levels impact the MGDA solution search by focusing on soft and hard decay rates for various variant datasets (small, medium, and large). In addition, the analysis of MGDA water level measurements is compared with the water level measurements from the previous adaptive GDA. The analysis is justified by the need to address the high costs caused by non-linearity, which isolates the cost function from the water level rate, necessitating a better understanding of how fixed parameters based on dataset size impact optimization. The hybridization of adaptive HSA with MGDA involves both evaluation and analysis stages. In the evaluation stage, the quality of the hybridization results is

compared to both adaptive HSA and state-of-the-art results. The goal is to assess the effectiveness of the hybridization approach. In the analysis stage, a fragment of the hybridization search space is examined to analyze acceptance threshold and pitch adjustment operator level gap.

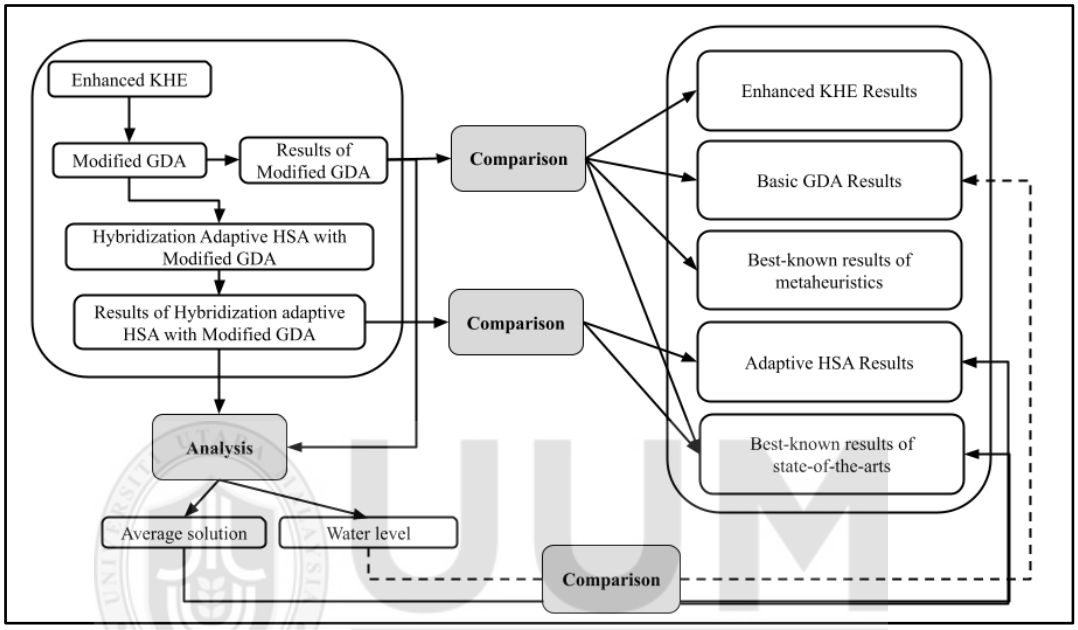


Figure 3.5. Evaluation and analysis of hybridization of adaptive HSA with modified GDA

The purpose is to highlight the strengths of the hybridization approach, including optimization, population diversity, and the ability to explore multiple solution paths.

3.5.1 Comparative Evaluation Criteria for Adaptive HSA Across Diverse Domains

The use of matrices for comparing various adaptations of HSA across selected criteria—scalability, robustness, efficiency, generalizability, innovation, and applicability—integrates a widely recognized and effective method within scholarly research into the specific context of optimization algorithms. Although direct citations explicitly stating this application may be more implicit, this analytical approach, deeply

rooted in the disciplines of computer science, engineering, and related fields, provides a structured and comprehensive means to systematically assess and delineate the relative strengths and weaknesses of each adaptive HSA variant. The framework outlined in this section is designed to facilitate a nuanced comparison that not only highlights the conceptual strengths and operational capabilities of each variant but also allows for meaningful analysis without necessitating additional experiments or simulations. Drawing from established practices in software metrics evaluation and optimization algorithm analysis, the principle behind employing such matrices is to enable clear, concise comparisons that support the decision-making process, tailored specifically to the intricacies and demands of **optimizing search exploration and exploitation balance** in diverse problem domains.

Scalability examines an adaptive HSA's capability to maintain or enhance performance as problem size or complexity increases. This involves evaluating how modifications in HSA parameters affect algorithmic efficiency across larger datasets or more complex optimization landscapes. Scalability is crucial for adaptive HSAs, especially in real-world applications where problems can be highly dynamic and scale significantly (Cormen et al., 2022). Scores range from 1 (significant performance drop) to 5 (maintains or improves performance with significant increases in problem size).

Robustness assesses the adaptive HSA's ability to deliver consistent performance despite uncertainties or variations in the problem domain, such as noisy data or fluctuating problem constraints. A robust adaptive HSA shows minimal performance degradation when faced with such challenges, underlining the algorithm's reliability

across diverse scenarios (Rousseeuw & Leroy, 2005). Scores range from 1 (high variability in performance) to 5 (minimal performance variability).

Efficiency pertains to the adaptive HSA's resource optimization, focusing on computational speed and memory usage. Efficiency is vital for ensuring that the algorithm can achieve optimal solutions within acceptable time frames and without excessive resource consumption, especially in resource-constrained environments (Bishop, 2006). Scores range from 1 (high resource usage and long computation times) to 5 (minimal resource usage and very fast computation times).

Generalizability refers to the adaptive HSA's versatility in solving a wide range of optimization problems without necessitating significant modifications to its parameters or structure. This criterion evaluates the algorithm's applicability across various domains, highlighting its capacity to adapt to different types of optimization challenges (Jordan & Mitchell, 2015). Scores range from 1 (effective on a very narrow range of problems) to 5 (highly versatile across a broad range of problems).

Innovation focuses on the novel aspects introduced by the adaptive modifications to the HSA, including new parameter tuning strategies or hybridization with other optimization techniques. This criterion assesses the algorithm's contribution to advancing the state of the art in optimization algorithms, particularly in how these innovations enhance performance or applicability (He et al., 2016). Scores range from 1 (minimal innovation) to 5 (highly innovative with substantial contributions).

Applicability considers the real-world relevance and implementation potential of the adaptive HSA in solving practical optimization problems. This includes evaluating the ease of integrating the algorithm into existing systems, its performance under real-world conditions, and its effectiveness in meeting diverse optimization needs (Kitchin, 2014). Scores range from 1 (limited applicability) to 5 (very high applicability with widespread real-world use).

To elucidate the application of the evaluation criteria, let's consider two adaptive Harmony Search Algorithms as shown in Section 2.5.1:

- Algorithm 1 (Mahdavi et al., 2007b) introduces a dynamic pitch adjustment rate (PAR) that changes with the generation number. It has been tested on various optimization problems, such as spring weight minimization and pressure vessel design, showing superior results compared to both the standard HSA and other heuristic approaches.

- Algorithm 2 (Wang & Huang, 2010) applies a decreasing PAR over time to prevent overshooting and oscillation. It incorporates a self-adaptive mechanism for parameter selection, tested on numerical optimization problems like the Sphere, Rosenbrock, and Ackley functions, demonstrating better performance than the basic HSA, enhanced HSA, and global-best HSA with 99% confidence.

The example comparison in Table 3.2 underlines the diverse strengths and specializations of adaptive HSAs, offering valuable insights into their suitability for various optimization problems. Algorithm 2 may be preferred for theoretical or

numerical optimization tasks requiring high adaptability and automated tuning. Conversely, Algorithm 1 might be more beneficial in applied scenarios with specific optimization goals, where its dynamic PAR adjustment offers targeted enhancements. By integrating this comparative analysis, the methodology section gains depth, illustrating the practical application of the evaluation framework in understanding and advancing optimization techniques across diverse field.

Table 3.2

Comparative Analysis of Algorithm 1 and Algorithm 2 in Optimization Tasks

Criterion	**Algorithm 1 (Mahdavi et al., 2007b)**	**Algorithm 2 (Wang & Huang, 2010)**
Scalability	Good scalability to complex, real-world problems through dynamic PAR. (Scores: 2)	Likely scalable across various problem sizes, effective in theoretical and simulation contexts. (Scores: 4)
Robustness	Adaptive robustness due to variation in PAR with generation number. (Scores: 3)	High robustness with a self-adaptive parameter mechanism, preventing instabilities. (Scores: 4)
Efficiency	Demonstrated efficiency, outperforming standard and heuristic approaches in specific tasks. (Scores: 3)	High efficiency, surpassing several variations of HSA with significant confidence. (Scores: 3)
Generalizability	Applied to real-world and constrained problems, showing generalizability in practical applications. (Scores: 4)	Strong generalizability within numerical optimization tasks. (Scores: 2)
Innovation and Applicability	Dynamic adjustment of PAR enhances innovation and applicability to diverse tasks. (Scores: 3.5)	Self-adaptive mechanism for parameter selection significantly enhances innovation and broad applicability. (Scores: 5)
Final Scores	Suited for practical engineering applications, offering tailored benefits in real-world scenarios. (Scores: 3.1)	Broader applicability due to the self-adaptive mechanism, enhancing efficiency, robustness, and generalizability. (Scores: 3.6)

The assessment acknowledges the diversity of optimization problems and seeks to identify overarching themes and capabilities that contribute to an adaptive HSA's success. Through this approach, the research explores the intricate nature of optimization algorithm performance and innovation, enhancing understanding of adaptive strategies in HSAs and their potential applications in complex problem-solving.

3.5.2 Dynamic Balance Factor

In this study, a dynamic balance factor is employed to manage the trade-off between exploration and exploitation during an iterative process, focusing on varying sizes of iterations (50, 100 and 200). The balance factor is calculated using a linear interpolation between two predefined weights: the exploration weight (α) set to 1.0, and the exploitation weight (β) set to 0.1 in order to measure the dynamic balance factor during each process of adaptive strategies in HSAs to determine whether these strategies can effectively handle exploration and exploitation. The exploration weight (α) is set to 1.0 to ensure that in the initial stages, the algorithm prioritizes exploring a wide range of possibilities. This high weight encourages the algorithm to investigate diverse solutions, which is critical for gathering comprehensive information about the search space and avoiding premature convergence to suboptimal solutions. Conversely, the exploitation weight (β) is set to 0.1 to shift the focus towards refining and improving the best-known solutions as the iterations progress. This lower weight ensures that, in the later stages, the algorithm concentrates on optimizing the solutions that have been identified as promising. The formula for the balance factor is:

$$Balance\ Factor = \alpha \left(1 - \frac{Current_{iteration}}{Max_{iteration}}\right) + \beta \left(\frac{Current_{iteration}}{Max_{iteration}}\right) \quad (3.4)$$

Where:

- α (set to 1.0) represents the initial exploration weight.
- β (set to 0.1) represents the initial exploitation weight.
- $Current_{iteration}$ is the current iteration number.
- $Max_{iteration}$ is the total number of iterations.

This balance factor transitions smoothly from emphasizing exploration in the early stages to focusing on exploitation in the later stages. At the beginning of the iterations, the balance factor is close to α, encouraging the algorithm to explore a wide range of possibilities and gather diverse information about the search space. As iterations progress, the balance factor decreases linearly, shifting the focus towards exploitation. This ensures that the algorithm concentrates on refining and improving the best-known solutions as it approaches the maximum number of iterations. This dynamic adjustment allows for a strategic balance between exploration and exploitation, enhancing the algorithm's ability to effectively navigate and optimize the solution space throughout the iterative process. Enhanced HSA will calculate its dynamic balance factor based on the above equation to compare with other existing adaptive HSAs, aiming to demonstrate improvements in performance and optimization efficiency.

For example, assume the total number of iterations $Max_{iteration}$ is 200, and the current iteration number $Current_{iteration}$ is 50. The exploration weight α is set to 1.0, and the exploitation weight β is set to 0.1.

The formula becomes:

$$Balance\ Factor = 1.0\left(1 - \frac{50}{200}\right) + 0.1\left(\frac{50}{200}\right)$$

Simplifying the terms inside the parentheses:

$$Balance\ Factor = 1.0\left(1 - \frac{50}{200}\right) + 0.1\left(\frac{50}{200}\right)$$

$$Balance\ Factor = 1.0(1 - 0.25) + 0.1\,(0.25)$$

$$Balance\ Factor = 1.0(0.75) + 0.1\,(0.25)$$

$$Balance\ Factor = 0.75 + 0.025$$

$$Balance\ Factor = 0.775$$

Thus, the Balance Factor is 0.775. This positive value indicates that exploration is still being prioritized, but there is also a small contribution from exploitation at this point in the process. As the current iteration (50) is still significantly smaller than the total number of iterations (200), the algorithm focuses more on discovering new areas (exploration) while gradually starting to incorporate some exploitation. As the iteration count increases, the Balance Factor will continue to shift, giving more emphasis to exploitation as the process progresses toward the total number of iterations, allowing for a strategic balance between exploration and optimization.

3.6 Summary

The proposed research methodology for optimizing HSTP comprises two phases. In the first phase, the initial school timetable constructed using the KHE algorithm, which then enhanced using the Markov Chain theory to model heuristic rules and improve the population of initial solutions. In the second phase, the basic HSA algorithm is implemented to improve the initial solution. The basic HSA was further enhanced in its parameters to boost better solutions. Next, to test the implementation of local search

algorithm for solving HSTP, the basic GDA is applied. The basic GDA is further improved in its water level for better solutions. Then both enhanced HSA and GDA were hybridized in order to produce more adaptive and effective solutions. In addition, the computational environment and results evaluation were applied to these two phases which also includes two evaluation metrics such as comparative evaluation criteria for adaptive HSA across diverse domains and dynamic balance factor. Moreover, dataset was produced from real-world school timetable. The dataset was solved using the construction and improvement phases developed in the first and second phases. Next Chapter describes the construction phase and its enhancement.

CHAPTER FOUR

MODELLING THE HEURISTIC RULES IN CONSTRUCTING SOLUTIONS FOR THE HIGH SCHOOL TIMETABLING PROBLEM

4.1 Introduction

This chapter focuses on addressing the objective one which is to model the heuristic rules in the time assignment stage for arranging the layer to enhance the KHE construction algorithm. In order to address this objective, firstly, the original KHE was implemented as explained in Section 4.2. The modelling of the heuristic rules in construction algorithms (KHE) is covered in this chapter including the experiment, results, and discussion. Section 4.3 illustrates step by step on producing the new model of heuristic rules. Next, Section 4.4 demonstrates the full experiment and results including the comparison with the state-of-the-art approaches. Section 4.5 analyses and discusses the proposed model. Finally, the final section concludes the Chapter.

4.2 Original KHE Implementation

KHE is an open-source ANSI C library that provides a fast and robust foundation for solving instances of HSTP expressed in the XHSTT format. It is intended for production use and research purposes and is released under the GNU public license.

Before installing the KHE, the build-essential package needs to be installed. The build-essential package is a collection of essential software development tools that contains the GCC (GNU Compiler Collection) which is required to compile C programs. In addition, specific C module called "KML" also need to be installed. This model consists

of a header file called "*kml.h*", as well as implementation files "*kml.c*" and "*kml_read.c*" that provides functionality for reading and writing XML files.

To install KHE, the libraries were downloaded and unpacked from the KHE website (http://jeffreykingston.id.au/khe/). The libraries contain a *makefile*, source files and documentation files. The *makefile* provides instructions on how to install and use the KHE. KHE has several data structures, including KHE_ARCHIVE, KHE_SOLN_GROUP, KHE_xs, KHE_TIME_GROUP, KHE_TIME, KHE_RESOURCE_TYPE, KHE_RESOURCE_GROUP, KHE_RESOURCE, KHE_EVENT_GROUP, KHE_EVENT, KHE_EVENT_RESOURCE, KHE_CONSTRAINT, KHE_SOLN, KHE_MEET, and KHE_TASK. These data structures represent different elements of a HSTP, such as instances, times, resources, events, constraints, solutions, and tasks. Users of KHE can read and write XML files, create solutions, add and change time and resource assignments using any algorithms they wish. Next, the benchmark datasets (XHSTT) were downloaded and archived in the same directory as the KHE folder.

All the data structures mentioned above were used in the four stages of KHE algorithm (structure, time assignment, resources assignment and clean-up stages) that explained in Figure 2.6 (Chapter 2). In the time assignment stage, the time were assigned in a specific heuristic rules follows:

1. **Index (*Ix*)**: it presents the original sort of layers based on the first stage of the KHE algorithm. This rule is the lowest priority compared to other rules.

2. **Demand (*De*)**: the meetings, which are inside one layer, are demanded from times to be assigned. This rule is higher than the index and lower than the duration.
3. **Duration (*Du*)**: it presents the highest priority compared to others.

4.3 Modification of KHE Original Heuristic Rules

This section explains on modification of KHE original heuristic rules to Markov Chain heuristic rules model (MCHR). MCHR is used to evaluate heuristic rules priority value. The flowchart (as shown in Figure 4.1) outlines a decision-making process for determining the sequence of visiting two layers. Firstly, two extra heuristic rules on top of duration (*Du*), index (*Ix*) and demand (*De*), are produced as the first decision value (Dv_1) and the second decision value (Dv_2) in order to manage the decision value for the layer. The Dv_1 is the combination of *De* and *Ix* which are less priority compared to the *Du*. The Dv_2 is the combination of *Du* and Dv_1which is the highest priority of decision for the layer in order to satisfy the same goal of the original heuristic rules.

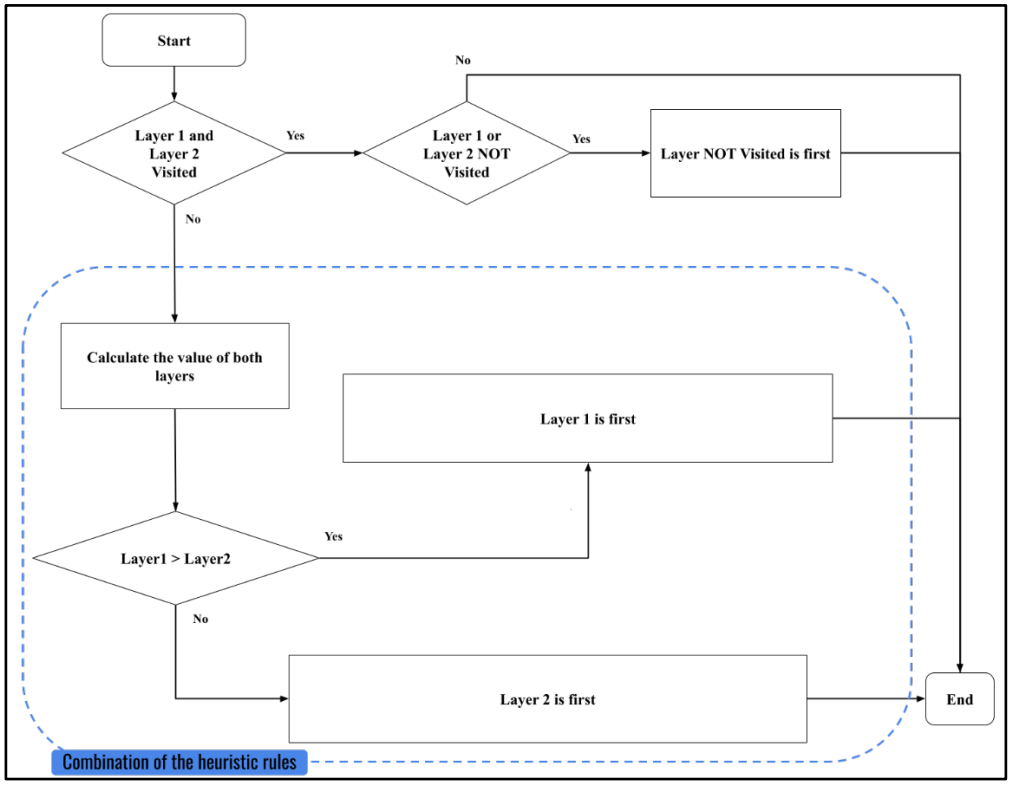


Figure 4.1. Flowchart after enhancing the sorting layers within the KHE time assignment phase

The values of *Ix, De* and *Du* need to be controlled to a value between 0 and 1 using Equation 2.7 (Chapter Two). Several considerations must be considered in order to understand why the values of *Ix, De* and *Du* must be controlled. Normalization, which first keeps measurements comparable while simplifying computations and lowering mistakes, ensures that results remain within a given range. These kinds of controls are also essential for computational system stability because some algorithms break down when their values move away from expectations. Moreover, values with a scale from 0 to 1 are frequently easier to understand, particularly when they represent relative measurements or probabilities. Finally, it is that these values must be controlled because the MCHR is especially made to function that best with inputs within this range. Due to that, the values of Dv_1 and Dv_2 will be generated as between 0 and 1. Equations 4.1, 4.2 and 4.3 show the formula of normalized values for *Ix, De* and *Du,* respectively.

These values which were generated during the process of layers sorting, were set as the input of MCHR.

$$Normalized\ Value\ of\ Ix = \frac{Current\ value\ of\ Ix - 1}{Size\ of\ Layer - 1} \tag{4.1}$$

$$Normalized\ Value\ of\ De = \frac{Current\ value\ of\ De - Minimum(De)}{Maximum(De) - Minimum(De)} \tag{4.2}$$

$$Normalized\ Value\ of\ Du = \frac{Current\ value\ of\ Du - Minimum(Du)}{Maximum(Du) - Minimum(Du)} \tag{4.3}$$

The five heuristic rules (as shown in Table 4.1) have their weightage values established by referring to the initial set of heuristic rules. In this process, β is used to determine the higher probability value of *De* compared to *Ix*, while $1-\beta$ sets the lower probability value of *Ix* compared to *De*. Similarly, α is used to establish the higher probability value of *Du* compared to Dv_1, while $1-\alpha$ determines the lower probability value of Dv_1 compared to *Du*. The concept of sorting layers which shown as connection and probability values (arrows) between the heuristic rules (circles) are shown in Figure 4.2. The heuristic rule circles of *Ix* and *De* that have probability values of $1-\beta$ and β, respectively, will generate the Dv_1 as calculated in Equation 4.4. Next, the heuristic rule circles of Dv_1 and *Du* that have probability values of $1-\alpha$ and α, respectively, will generate the Dv_2 as calculated in Equation 4.5.

Table 4.1

Weightage Values

Concept	Dv_2	Dv_1
Dv_2	-	-
Du	α	-
Dv_1	$1-\alpha$	-
De	-	β
Ix	-	$1-\beta$

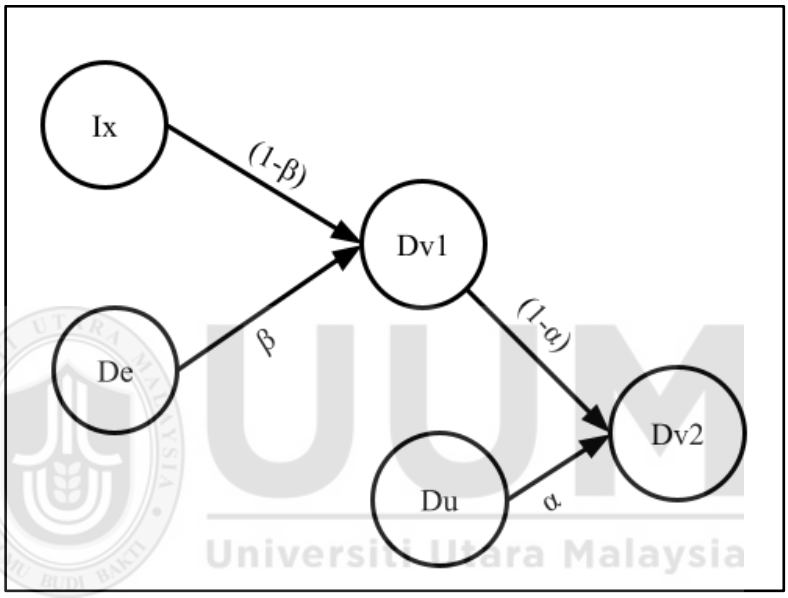


Figure 4.2. Conceptual of sorting layers

$$Dv_1 = (1-\beta)\,.Ix + \beta.De \tag{4.4}$$

$$Dv_2 = \alpha.Du + (1-\alpha).Dv_1 \tag{4.5}$$

All calculations of MCHR (Equations 4.1 until 4.5) were executed in the process of *Calculate the value of both layers* in the Figure 4.1. Next, the layer that have higher value of Dv_2 is sorted first and the times will be assigned to meets of the layer. The process of layer sorting will be repeated until all layers are assigned with times.

The MCHR Decision Process algorithm pseudocode (as shown in Figure 4.3) outlines the decision process for the MCHR. It takes two inputs, Layer1 and Layer2, and begins by checking if both layers have been visited. If both are visited, the process returns immediately without any further action. If either one or both layers have been visited but not both, it checks whether Layer1 has not been visited. If Layer1 has not been visited, the algorithm proceeds to process Layer1. Otherwise, it processes Layer2. After handling the visitation condition, the algorithm proceeds to calculate a value for both layers using equations referred to as Equations 4.1 through 4.5. These equations presumably calculate specific characteristics of the layers. Once the values for both layers are calculated (valueLayer1 and valueLayer2), it compares the two.

Algorithm 4.1. MCHR Decision Process

Input: $Layer1, Layer2$

```
BEGIN
    IF Layer1 Visited AND Layer2 Visited THEN
        RETURN
    END IF
    IF Layer1 Visited OR Layer2 Visited THEN
        IF Layer1 NOT Visited THEN
            Process Layer1
        ELSE
            Process Layer2
        END IF
        valueLayer1 = CalculateValue(Layer1) using Equations 4.1 until 4.5
        valueLayer2 = CalculateValue(Layer2) using Equations 4.1 until 4.5
        IF valueLayer1 > valueLayer2 THEN
            Process Layer1
            Process Layer2
        ELSE
            Process Layer2
            Process Layer1
        END IF
END
```

Figure 4.3. Pseudocode for algorithm MCHR decision process

If valueLayer1 is greater than valueLayer2, it first processes Layer1 followed by Layer2. If valueLayer1 is not greater, the process is reversed, with Layer2 being processed first, followed by Layer1. This structured decision-making process likely helps in determining an efficient sequence or priority of tasks (layers), ensuring that unvisited layers or those with higher calculated values are given preference in processing order. The overall flow relies on checking visitation status and computed values to determine the order of processing for the layers.

4.4 Experiment and Results

To run the original KHE in terminal, the "*make khe*" comment was executed to produce a binary executable files called "*khe*" that includes the *main*() function defined in the "*main.c*" file. KHE has a function called "*KheParallelSolve()*" that is designed to solve multiple solutions in parallel. This function will call the *KheArchiveParallelSolve* function with the prepared archive solver options. This *KheArchiveParallelSolve* function parallelizes the solving of instances in the archive using a specified solver options, such as varying the size of the population (e.g., 8, 16, 32, 64, 128, 256, or 512). The solver options allow for trying out different population sizes for each dataset, based on experimental results to determine the best value as shown in Table 4.2. The solutions will be saved in the solution group, if specified, and other memory management operations (such as NS and sorting) will be performed as specified by the options.

Table 4.2

Population number used that produce the best population of initial solutions for each XHSTT dataset

Country	Data Instances	Category	Population number used that produce the best population of initial solutions
Australia	"AU-BG-98"	M	128
	"AU-SA-96"	M	128
	"AU-TE-99"	M	8
Brazil	"BR-SA-00"	S	128
	"BR-SM-00"	S	128
	"BR-SN-00"	S	128
	"BrazilInstance5"	S	128
	"BrazilInstance3"	S	128
	"BrazilInstance7"	S	128
	"BrazilInstance1"	S	128
Czech	"VillageSchool"	S	16
Denmark	"DK-FG-12"	L	8
	"DK-HG-12"	L	8
	"DK-VG-09"	L	8
Spain	"ES-SS-08"	S	32
Finland	"FI-PB-98"	M	128
	"FI-WP-06"	S	128
	"FI-MP-06"	M	128
	"SecondarySchool2"	M	128
	"ElementarySchool"	M	128
	"ArtificialSchool"	M	128
Greece	"GR-H1-97"	M	128
	"GR-P3-10"	S	128
	"GR-PA-08"	M	128
	"UniversityInstance3"	S	128
	"UniversityInstance5"	S	128
	"Preveza2008"	M	128
	"Aigio2010"	M	128
Italy	"IT-I4-96"	M	32
Kosovo	"KS-PR-11"	L	32
Netherlands	"NL-KP-03"	L	8
	"NL-KP-05"	L	8
	"NL-KP-09"	L	8
	"Kottenpark2008"	L	8
	"GEPRO"	L	8
UK	"UK-SP-06"	L	8
USA	"US-WS-09"	M	128
South Africa	"ZL-LW-09"	S	128
	"ZA-WD-09"	M	128

The setting of MCHR model consist of β and α that are firstly set based on two values which are 0 and 1 as shown in Table 4.3. The setting 1 produces decision value Dv_2 that equal to Dv_1 which prioritized layers based on index (Ix). The setting 2 produces decision values Dv_2 that equal to Dv_1 which prioritized layers based on demand (De). The settings 3 and 4 produce decision value Dv_2 which prioritized layers based on duration (Du).

Table 4.3

The setting of β and α for MCHR Model

Setting	β	α	Equation 4.4 $(Dv_1 = (1-\beta)\,.Ix + \beta.De)$	Equation 4.5 $(Dv_2 = \alpha.Du + (1-\alpha).Dv_1)$
1	0	0	$Dv_1 = (1-0)\,.Ix + 0.De$ $Dv_1 = Ix$	$Dv_2 = 0.Du + (1-0).Dv_1$ $Dv_2 = Dv_1$
2	1	0	$Dv_1 = (1-1)\,.Ix + 1.De$ $Dv_1 = De$	$Dv_2 = 0.Du + (1-0).Dv_1$ $Dv_2 = Dv_1$
3	0	1	$Dv_1 = (1-0)\,.Ix + 0.De$ $Dv_1 = Ix$	$Dv_2 = 1.Du + (1-1).Dv_1$ $Dv_2 = Du$
4	1	1	$Dv_1 = (1-1)\,.Ix + 1.De$ $Dv_1 = De$	$Dv_2 = 1.Du + (1-1).Dv_1$ $Dv_2 = Du$

The four settings above show that setting of β and α to 0 or 1 are actually implemented the original heuristic rule. From this point of view based on the of preliminary experiments, the suitable setting of β and α for MCHR model are set as 0.99. There are various reasons for the MCHR model's decision to select 0.99 for both β and α. As the value is close to 1 which suggesting possible adjustment for better performance. Absolute values, like 0 or 1, may lead to extreme behaviors or unstable in models. For this reason, 0.99 ensures that factors of Ix, De, and Du receive fair attention. Preliminary experiments suggesting this value's superiority over alternative settings may have led to its empirical formulation. Additionally, 0.99 encourages a balanced input of Ix, De, and Du which ensures that no single factor dominates or "overshadows"

the others. Finally, 0.99 might provide a degree of flexibility to the model, improving its capacity to adjust to a variety of situations or data sets.

Table 4.4 shows the results of original KHE and MCHR model based on 39 datasets. The results are also compared with lower bound. The results show that the MCHR model can improve the solutions generated by the original KHE in 25 out of 39 data instances (64% of improvement), while produces similar results in five datasets (*GR-H1-97, Aigio2010, UniversityInstance3*, *ElementarySchool* and *UniversityInstance5*). The best performance of the MCHR model is on *AU-TE-99* dataset which all hard constraints are reduced to zero and the soft constraints are decreased to 87 from 152 in the original KHE. The MCHR model can even minimize the soft constraints violations in *AU-BG-98* datasets (from 584 to 557) and *DK-VG-09* datasets (from 4100 to 3993), respectively. However, the static nature of the MCHR model might be the reason for several datasets are not improved such as *BR-SN-00, BrazilInstance7, FI-WP-06, FI-MP-06, US-WS-09 GEPRO, NL-KP-05, NL-KP-03,* and *GR-PA-08* datasets.

To compare between original KHE and MCHR model, the average rank is used. It is found that MCHR model obtained average of 2.64 comparing 3.26 of KHE. MCHR model is found to be very competitive with KHE and able to achieve some best solutions. Moreover, MCHR achieves optimal solution in the *GR-P3-10* dataset.

Table 4.4

Results Found Utilizing the MCHR model Compared to KHE

Instance	Lower Bound	Construction Algorithms	
		KHE	MCHR Model
AU-BG-98	0.00000	1.00584	1.00557
AU-SA-96	0.00000	0.00018	0.00003
AU-TE-99	0.00020	2.00152	0.00087
BR-SA-00	0.00005	0.00014	0.00011
BR-SM-00	0.00051	3.00123	2.00125
BR-SN-00	0.00035	0.00089	0.00096
BrazilInstance5	0.00019	0.00085	0.00083
BrazilInstance3	0.00024	0.00074	0.00065
BrazilInstance7	0.00040	0.00123	0.00132
BrazilInstance1	0.00041	0.00049	***0.00048***
VillageSchool	0.00010	0.00014	***0.00013***
DK-FG-12	0.00412	0.01794	0.01678
DK-HG-12	0.00000	12.04637	12.04582
DK-VG-09	2.00000	2.04100	2.03993
ES-SS-08	0.00000	0.00616	0.00565
FI-PB-98	0.00000	0.00008	0.00005
FI-WP-06	0.00000	0.00012	0.00014
FI-MP-06	0.00077	0.00090	0.00091
SecondarySchool2	0.00000	0.00004	***0.00002***
ElementarySchool	0.00003	***0.00003***	***0.00003***
ArtificialSchool	0.00000	4.00005	3.00004
GR-H1-97	0.00000	***0.00000***	***0.00000***
GR-P3-10	0.00000	0.00002	***0.00000***
GR-PA-08	0.00003	0.00005	0.00006
UniversityInstance3	0.00005	0.00006	0.00006
UniversityInstance5	0.00000	0.00000	***0.00000***
Preveza2008	0.00000	0.00004	***0.00002***
Aigio2010	0.00000	0.00008	***0.00008***
IT-I4-96	0.00027	0.00040	0.00039
KS-PR-11	0.00000	0.00012	0.00009
NL-KP-03	0.00000	0.00787	0.00899
NL-KP-05	0.00089	0.01019	3.01142
NL-KP-09	0.00000	10.05125	8.06360
Kottenpark2008	0.02795	22.24125	***14.26838***
GEPRO	1.00000	1.00905	1.01683
UK-SP-06	0.00000	28.01092	***13.00880***
US-WS-09	0.00101	0.00532	0.00537
ZL-LW-09	0.00000	13.00016	2.00034
ZA-WD-09	0.00000	16.00000	2.00000
Average Rank	**-**	**3.26**	**2.64**
Rank	**-**	**2**	**1**

4.5 Results Analysis and Discussion

As shown in Figure 4.4, the MCHR model is able to rapidly decrease with equal and less than 0.50 level gap value of both hard and soft constraints for seven datasets (such as *AU-TE-99, AU-SA-96, SecondarySchool2, Preveza2008, GR-P3-10, UK-SP-06* and *ZA-WD-09*). The calculation of the level gap is explained in Section 3.6. Five datasets are decreased with a 0.25 gap level *(such as BR-SM-00, VillageSchool, FI-PB-98, NL-KP-09* and *Kottenpark2008*). The rest of the datasets are slightly decreased with a 0.10 level gap. Figure 4.4 showcases with significance difference of hard and soft constraint gap levels between original KHE and MCHR in six datasets, namely *BR-SM-00, ArtificialSchool, NL-KP-09, Kottenpark2008, UK-SP-06,* and *ZL-LW-0*. However, the largest dataset called *GEPRO* dataset is unable to decrease the gap level of soft constrains. The same situation goes for another seven datasets such as *NL-KP-05, NL-KP-03, GR-PA-08, FI-MP-06, FI-WP-06, BrazilInstance7* and *BR-SN-00.*

Based on Figure 4.3, the MCHR model optimally improved the hard constraints while decreasing half of the soft constraint gap levels for *AU-TE-99*. Table 4.5 shows a comparison of order layers between KHE and the MCHR model for *AU-TE-99*. From the table, the KHE model indicates that layer N (the values of *Du* (17) and *De* (127)) wil be the last layer to be assigned. However based on the MCHR, the smallest values of *Du* (6) and *De* (18) will be assigned as the last layer. Based on the suggestion by the MCHR model, it is found that the violated constraints are less than the solution produced by the KHE. As clearly seen in Table 4.5, the order of layers based on *Du* and *De* values give better results of initial solutions when it is well sorted.

To address how the KHE problem is solved by MCHR, it is important to understand the improvements that MCHR brings over KHE. The KHE algorithm operates by using a fixed set of heuristic rules—duration (Du), demand (De), and index (Ix)—to assign times and resources in a high school timetabling problem. However, this static approach has its limitations in prioritizing layers and adapting to different datasets.

The MCHR model enhances the original KHE by introducing a dynamic decision-making process. It modifies the heuristic rules by creating two decision values, Dv1 and Dv2. These decision values are combinations of the original heuristics. Dv1 is calculated from demand and index, while Dv2 incorporates duration and Dv1. By layering these decisions, MCHR refines the order of time assignments, ensuring that the more complex, higher-priority layers are processed first. This hierarchical sorting process ensures a more optimal assignment of time slots compared to the static priority rules used by KHE.

Another key improvement of MCHR is the normalization of heuristic values between 0 and 1. This normalization not only simplifies the calculations but also enhances computational stability by preventing extreme or unexpected value ranges from disrupting the algorithm. The normalized values also make it easier to compare different layers based on relative importance, further refining the layer-sorting process.

MCHR introduces a dynamic decision-making mechanism in contrast to KHE's static approach. It calculates the priority of each layer in real-time, based on the heuristic values, and assigns time slots to the most important layers first. This process is repeated iteratively until all layers are assigned, ensuring that the assignments are always

optimized. The experimental results show that MCHR produces better solutions than KHE, improving outcomes in 25 out of 39 datasets.

The practical impact of MCHR is significant, as it resolves the limitations of the KHE algorithm by reducing both hard and soft constraint violations. For example, in the AU-TE-99 dataset, MCHR significantly reduced the number of soft constraint violations compared to KHE. This improvement demonstrates that MCHR provides a more flexible, efficient, and optimized solution for high school timetabling problems, solving the issues present in KHE by dynamically adjusting the order of time assignments.

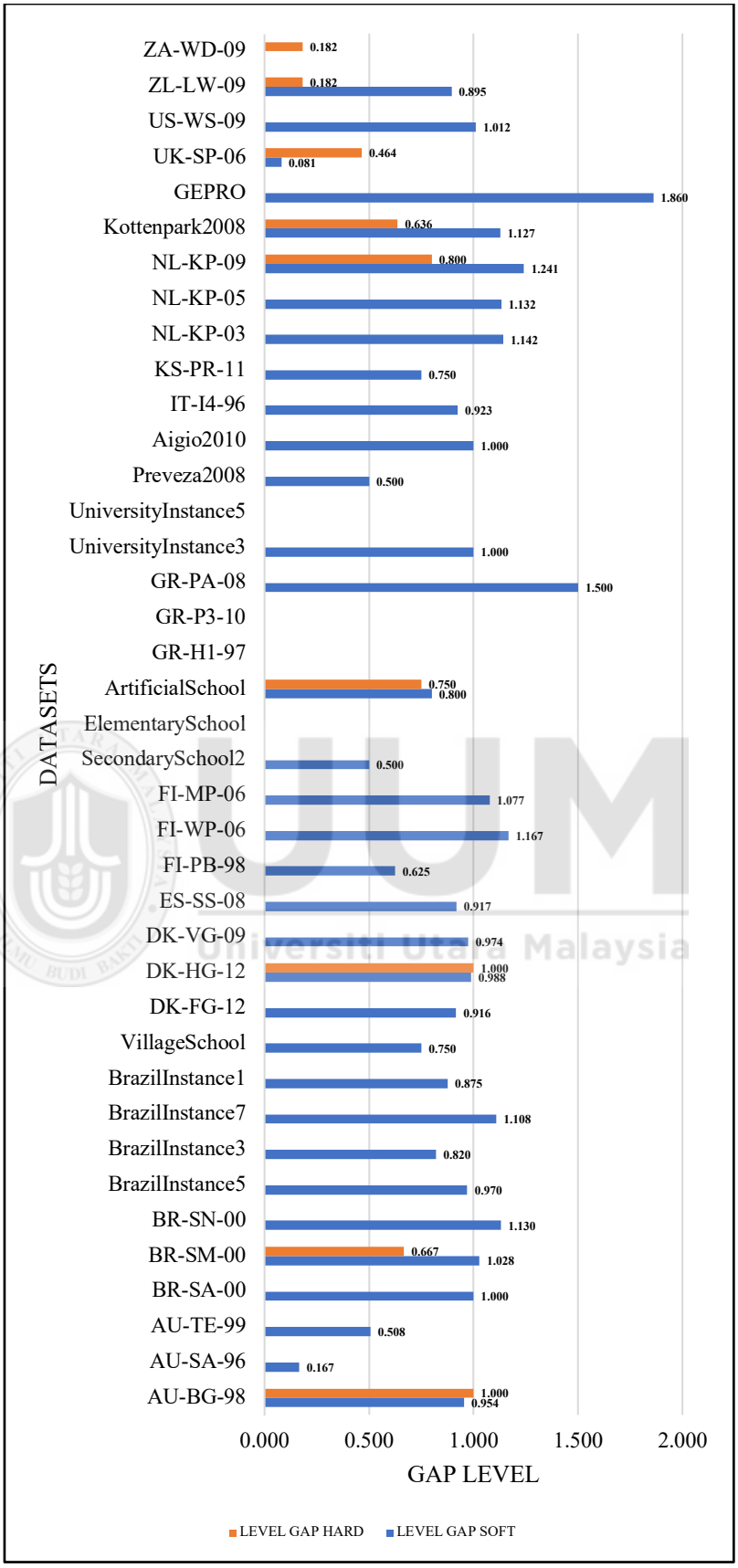


Figure 4.4. Hard and soft constraints level gap between MCHR model and original KHE for 39 datasets

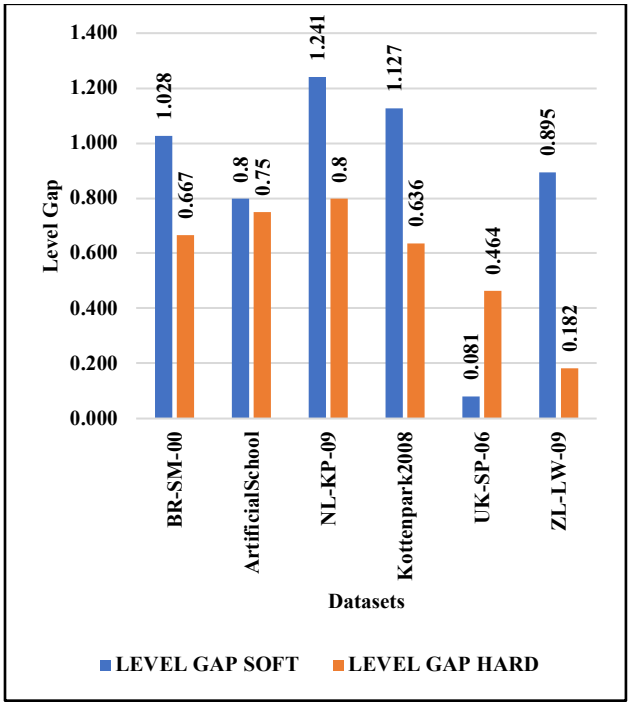


Figure 4.5. The gap level of hard and soft constraints of six datasets

Table 4.5

Comparison of Order Layers Between KHE and the MCHR Model for AU-TE-99

#	After Time Assignments Phase			
	KHE		MCHR	
Layers	Du	De	Du	De
Layer 1	30	224	30	302
Layer 2	**6**	**18**	30	224
Layer 3	14	163	30	224
.	.	.	.	.
.	.	.	.	.
Layer N	17	127	**6**	**18**

4.6 Summary

Within the time assignment phase of the KHE algorithm, this chapter introduces a sorting strategy based on the Markov Chain model. The MCHR model is a static model that calculates the priority value of layers by associating numerous layer properties such

as demand, duration, and index. When compared to KHE, the MCHR model is able to produce better HSTP solution outcomes on the majority of the XHSTT dataset. However, when compared to KHE, certain XHSTT instances were unable to improve their solution, which could be due to the static nature of MCHR. The findings of this Chapter can be coupled with population-based metaheuristics to solve the HSTP as first solutions in the improvement phase in the next Chapter.

CHAPTER FIVE

ADAPTIVE HARMONY SEARCH ALGORITHM FOR SCHOOL TIMETABLE

5.1 Introduction

This chapter addresses the research objective of making adaptive adjustments to the harmony memory consideration rate (HMCR), random consideration rate (RCR), and pitch adjustment rate (PAR) based on (1) iteration position and solution number, (2) behavioural status, and (3) parameters linkages. To address the mentioned research objective, this chapter describes the improvement phase to solve the HSTP using basic HSA and enhancement HSA. Firstly, the basic HSA is implemented with fixed parameters. Next, an enhanced HSA with adaptive parameters (that produce parameter values based on iteration position and solution number, behavioural status, and parameters linkages) is carried out. Experiments are conducted to test both fixed and adaptive HSA in finding solutions for solving the HSTP. The analysis and discussion of the adaptive HSA is provided in the following section. Final section presents the summary of the chapter.

5.2 Basic HSA Implementation (Fixed HSA)

In this section, the basic HSA algorithm as explained in Section 2.5, is implemented with modification on RCR and PAR which divided into several sub-parameters based on number of NS as shown in Figure 5.1 and further details using flowchart as presented in Figure 5.2.

Figure 5.2 presents the flowchart of the basic HSA. In step 1, the initial set of solutions is obtained using enhanced KHE algorithm known as MCHR as explained in Chapter Four, which is represented by the GroupSolutions and its size (HMS).

Algorithm 5.1 Fixed HSA
Input: $GroupSolutions, NS, MI$
Output: $GroupSolutions$

```
While t < MI Do
    Rand ← rand(0,1)
    /* Harmony Memory Consideration (HMC)*/
    if HMCR ≤ Rand then
        /*Pitch Adjustment (PA)*/
        PAR_rate ← rand(0,1)
        if PAR_rate ≤ 1 AND PAR_rate > PAR1 then
            NS_MeetsSwap
        else if PAR_rate ≤ PAR1 AND PAR_rate > PAR2 then
            NS_TasksSwap
        else if PAR_rate ≤ PAR2 AND PAR_rate > PAR3 then
            NS_MeetsMoveTime
        else if PAR_rate ≤ PAR3 AND PAR_rate > PAR4 then
            NS_TasksMoveResource
        else if PAR_rate ≤ PAR4 AND PAR_rate > PAR5 then
            NS_BlockMeetsSwap
        else
            NS_KempeMeetMoveTime
        End if
    Else
        /* Random Consideration (RC)*/
        RCR_rate ← rand(0,1)
        if RCR_rate ≤ 1 AND RCR_rate > RCR1 then
            NS_KempeMeetMove
        else if RCR_rate ≤ RCR1 AND RCR_rate > RCR2 then
            NS_TaskEjectingMoveResource
        else if RCR_rate ≤ RCR2 AND RCR_rate > RCR3 then
            NS_EjectingMeetMoveTime
        else
            NS_EjectingMeetMove
        End if
    End if
End while
return GroupSolutions
```

Figure 5.1. Pseudocode for harmony improvisation of fixed HSA

In step 2, several parameters are set, including the NS, HMCR, RCR, PAR, MI and NS.

The best values for HMCR, RCR, PAR and MI are set based on the number of NS and preliminary experimentation. For the fixed parameters of HSA, the new harmony solution's values are picked at random from the ones currently stored in GroupSolutions with a probability of HMCR where HMCR $\in [0,1]$. Decision variables are chosen randomly in RCR based on their potential range with a probability of (1-HMCR). RCR and PAR will be divided into several sub-parameters based on number of NS such as RCR1, RCR2, RCR3, PAR1, PAR2, PAR3, PAR4 and PAR5 that represent values between 0 and 1 (in Figure 5.1, from line 7 to line 19 for PAR and from line 23 to line 31 for RCR). These parameters would be tuned based on the specific characteristics of the HSTP and the optimization goals.

Random value is generated between 0 and 1 in step 4. If the HMCR value, that was set in step 2, is greater or equal to the generated random value, the step 5 will generate another random value that assign to PAR_rate variable (pitch adjustment operator). Otherwise, the random consideration operator will be executed. The PAR_rate value will be checked whether it is less than or equal to PAR1 value (set in step 2). If yes, the first NS will be executed. Otherwise, the rest of NS condition statements will be executed. Based on the results of this initial experiment, the NS sequence has been selected based on effective solutions generation. In random consideration operator, it will generate a new random value that will be assigned to RCR_rate (step 12). Furthermore, it will execute the same conditional statements as the pitch adjustment operator.

The HMCR and RCR define the rates at which values are selected from the harmony memory or generated randomly for improvisation. The PAR defines the rate at which values are adjusted during improvisation. The MI defines the maximum number of iterations to perform, and the HMS defines the number of harmonies to store in the harmony memory. After setting the parameters, the algorithm generates an initial population or harmony memory based on the HMS. Each harmony is a set of values for the decision variables in the HSTP. The values are randomly generated within the feasible range of the HSTP. In each iteration of the HSA, a new harmony is generated by randomly selecting a row from the harmony memory and improvising it. In the HSA, three operators are used for harmony improvisation: memory consideration (MC), random consideration (RC), and pitch adjustment (PA).

The new harmony is evaluated using the second type of objective function as explained in Section 2.3, and if it is better than the worst harmony in the harmony memory, it is added to the memory in place of the worst harmony. The algorithm stops when it reaches the maximum number of iterations (MI). Finally, the best solution found in the harmony memory is returned as the final solution as known Cadenza.

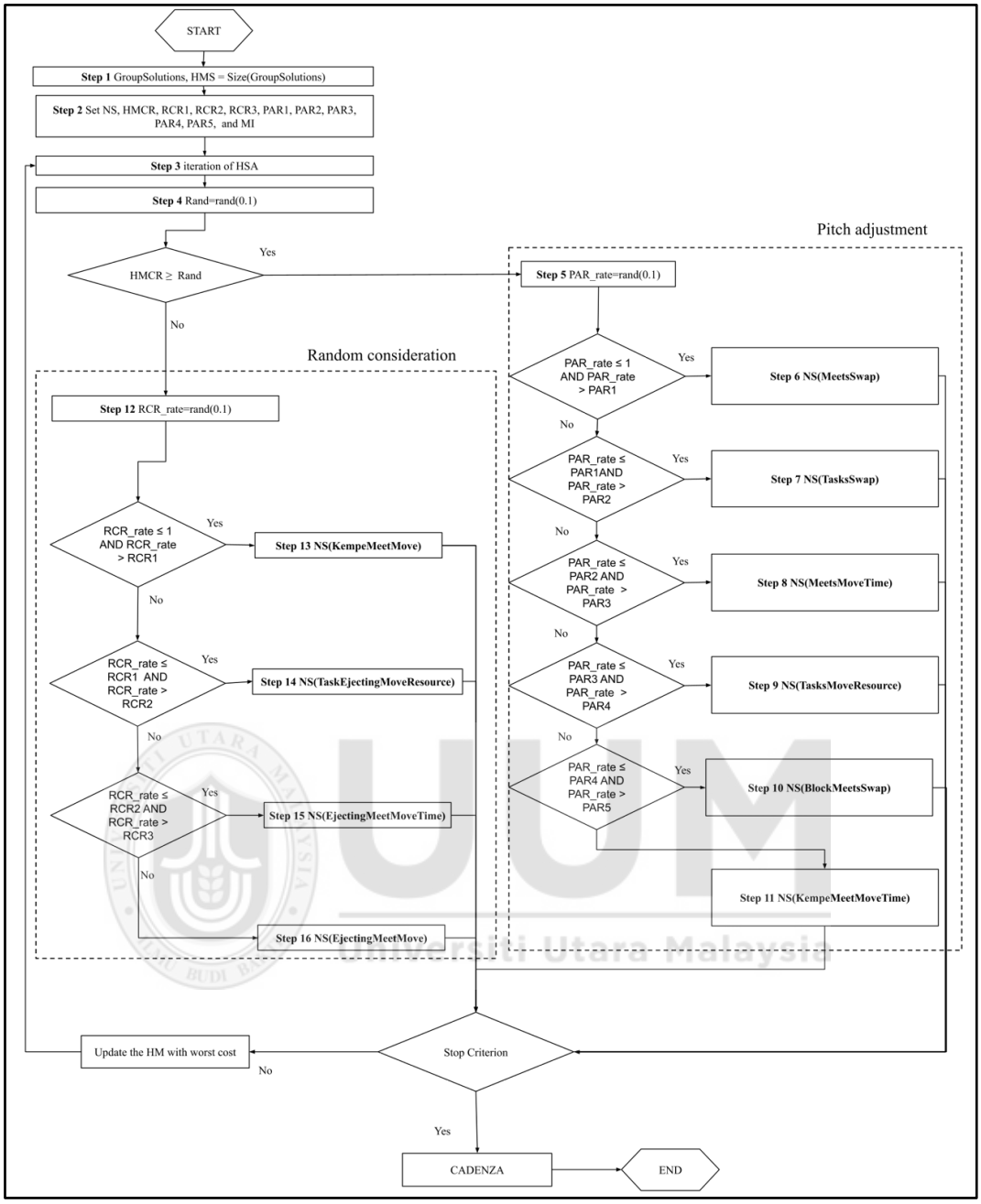


Figure 5.2. Basic HSA flowchart

5.3 Adaptive Harmony Search Algorithm (DHSA)

In this section, the proposed adaptive HSA as shown in pseudocode (Figure 5.3) and flowchart (Figure 5.4), is described on adaptive adjustment of harmony memory consideration rate (HMCR), random consideration rate (RCR), and pitch adjustment

rate (PAR) based on (1) iteration position and solution number, (2) behavioural status, and (3) parameters linkages.

Algorithm 5.2 DHSA
Input: $GroupSolutions, NS, MI$
Output: $GroupSolutions$

```
While t < MI Do
    Rand ← rand(0,1)
    RCR ← Equation 5.1
    HMCR ← 1 − RCR
    /* Harmony Memory Consideration (HMC)*/
    if HMCR ≤ Rand then
        /*Pitch Adjustment (PA)*/
        PAR_rate ← rand(0,1)
        if SizeOf(equal_improve) > SizeOf(improve) then
            PAR ← 1 − Equation 5.2
        else
            PAR ← Equation 5.2
        End if
        if PAR_rate ≤ PAR then
            rand (NS_i ⇒ NS_i = improve)
        else
            rand (NS_i ⇒ NS_i = equal_improve)
        End if
    Else
        /* Random Consideration (RC)*/
        rand (NS_i ⇒ NS_i = non_improve)
    End if
End while
return GroupSolutions
```

Figure 5.3. Pseudocode for DHSA

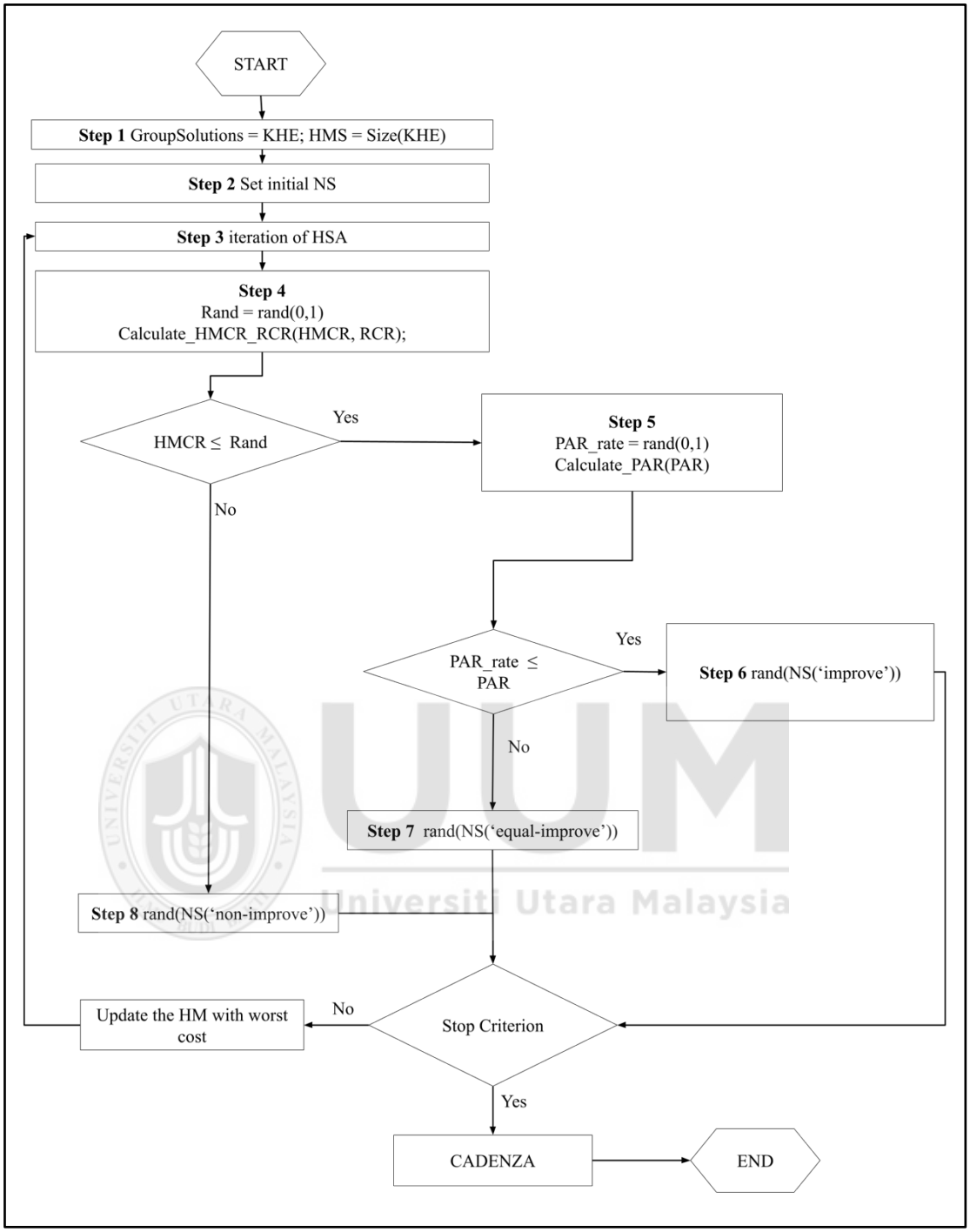


Figure 5.4. DHSA flowchart

Figure 5.4 presents the flowchart of the DHSA. In step 1, the initial set of solutions is obtained using enhanced KHE algorithm known as MCHR as explained in Chapter Four, which is represented by the GroupSolutions and its size (HMS). In the step 2, all the solution of each NS is initially set to **"improve"** status (each solution could have

either one of the three statuses, which are **"improve," "equal-improve"** or **"non-improve**"). The status is given to each solution in the HM after each iteration of HSA related to the results from the neighborhood move as shown in Figure 5.5. The status is updated after the NS is applied to a solution, thereby helping the DHSA change the setting of the parameters (HMCR, RCR, and PAR) depending on the NS status and solution.

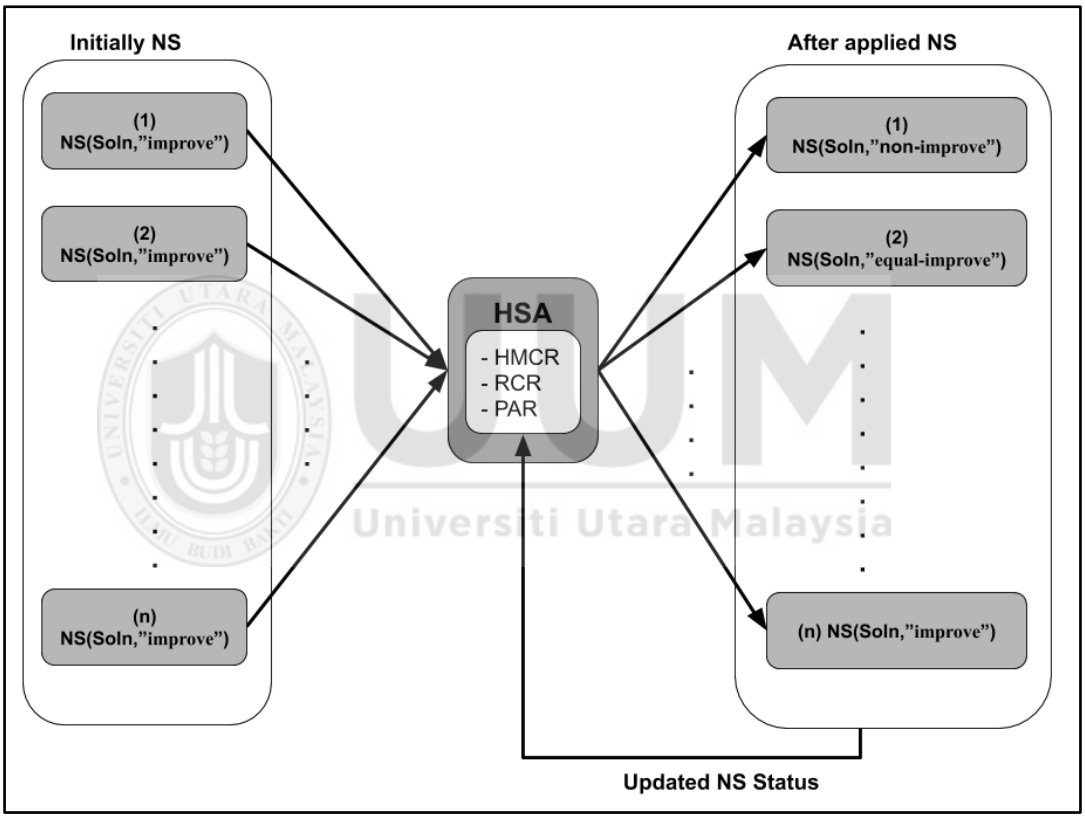


Figure 5.5. Initial and updated status of NS after iteration for DHSA

Step 3 is the outer loop of DHSA iteration. In step 4, there are three sub steps, i.e., 1) the random value (Rand) is generated to be compared later with HMCR value, 2) calculation of RCR adaptively using Equation 5.1, and 3) calculation of HMCR (1 - RCR).

$$RCR(t,s) = \beta \times \left\{ \alpha \times \left(\frac{\sum_{\text{Improve}=1}^{n} NS_{\text{Improve}}(t,s)}{n} \right) + (1-\alpha) \times \left(\frac{\sum_{\text{EqualImprove}=1}^{n} NS_{\text{EqualImprove}}(t,s)}{n} \right) \right\} + (1-\beta) \times \left(\frac{\sum_{\text{Nonimprove}=1}^{n} NS_{\text{Nonimprove}}(t,s)}{n} \right) \quad (5.1)$$

Where:

- ***t*:** it is the value of iteration factor.
- ***s*:** it is the value of solution position.
- ***Improve*:** It is neighbourhood move position where Improve = 1, 2, .., n.
- ***EqualImprove*:** It is neighbourhood move position where EqualImprove = 1, 2, .., n
- ***Nonimprove***: It is neighbourhood move position where Nonimprove = 1, 2, .., n
- ***n*:** it is the size of NS.
- ***α*:** it is the contribution value based on the previous mechanism of setting for HMCR and RCR.
- ***β*:** it is the contribution value based on the previous mechanism of setting for HMCR and RCR.

For sub step 2, in Equation 5.1, the value of α and β are set based on the previous basic HSA setting of *HMCR* and *RCR*. In most studies, the *HMCR* value is set to always be higher than the *RCR* value (Al-Betar & Khader, 2012; Shambour et al., 2013; Wahid & Hussin, 2014). Regarding to NS solution status (**"improve," "equal-improve"**, **"non-improve**"), the NS with **"improve"** status has higher priority compared to NS with **"equal-improve"** status in the search space. While, both NS with **"improve"** and **"equal-improve"** statuses are higher priority compared to NS with "**non-improve**"

status in the search space. Based on this, the total number of $NS(t,s)$ that have **"improve"** status is divided by n and multiplied by α to give more possibility of occurrence on the **"improve"** status. Next, total number of $NS(t,s)$ that have **"equal-improve"** is divided by n and multiplied by $1-\alpha$ to reduce the possibility of occurrence on the **"equal-improve"** status. Both of possibility values of NS with **"improve"** and **"equal-improve"** statuses are multiplied by β to give more possibility of occurrence on the NS with **"improve"** and **"equal-improve"** statuses status. Meanwhile, the total number of $NS(t,s)$ that have **"non-improve"** status is divided by n and multiplied by $1-\beta$ to reduce the possibility of occurrence on the **"non-improve"** status.

This Equation 5.1 is applied only if the total number of NS with **"improve"** and **"equal-improve"** statuses (the sum of **"improve"** and **"equal-improve"** statuses) is greater than 0 and the number of NS with **"non-improve"** is greater than 0. Noted that if the number of NS that have **"improve"** and **"equal-improve"** statuses are equal to 0, then the *RCR* is given higher priority (set with value 1) than the *HMCR* (set with value 0), indicating that all NS cannot be improved. If the number of NS that have **"non-improve"** status is 0, then the *HMCR* is given higher priority (set with value 1) than the *RCR* (set with value 0), indicating that all NS can be improved or have the same solution. For sub step 3, the calculation of HMCR is produced by minus the RCR with 1 (1 - RCR). Next, the value of HMCR will be compared with the random value (Rand) that produced earlier. If the value of HMCR is less or equal to Rand, step 5 will be executed. Otherwise, step 8 (explained later) will be executed. In step 5 (line 8 until 13 in Figure 5.3), there are two sub steps, i.e., 1) the random value (*PAR_rate*) is generated

to be compared later with PAR value, and 2) calculation of PAR adaptively using Equation 5.2 followed by calculation of 1 – PAR.

$$PAR(t,s) = \gamma \times \left(\frac{\sum_{Improve=1}^{n} NS_{Improve}(t,s)}{\{\sum_{EqualImprove=1}^{n} NS_{EqualImprove}(t,s) + \sum_{Improve=1}^{n} NS_{Improve}(t,s)\}} \right) + (1-\gamma) \times \left(\frac{\sum_{EqualImprove=1}^{n} NS_{EqualImprove}(t,s)}{\{\sum_{EqualImprove=1}^{n} NS_{EqualImprove}(t,s) + \sum_{Improve=1}^{n} NS_{Improve}(t,s)\}} \right) \quad (5.2)$$

Where:

- ***t*:** it is the value of iteration factor.
- ***s*:** it is the value of solution position.
- ***Improve*:** It is neighbourhood move position where Improve = 1, 2, .., n.
- ***EqualImprove*:** It is neighbourhood move position where EqualImprove = 1, 2, .., n
- ***n*:** it is the size of NSs.
- ***γ*:** it is the contribution value based on the previous mechanism of setting for PAR and neighbourhood moves.

For sub step 2, if total number of NS with "**equal-improve"** status is greater than the total number of NS with **"improve"** status, the PAR value will be calculated by 1 minus with the Equation 5.2. Otherwise, the Equation 5.2 will be calculated and assigned to PAR. In Equation 5.2, as the NS with **"improve"** status has higher priority compared to NS with **"equal-improve"** status in the search space, the total number of $NS(t,s)$ that have **"improve"** status is divided by the sum of NS with **"improve"** and **"equal-improve"** statuses and multiplied by γ to give more possibility of occurrence on the

"improve" status. Noted that, the reason of addition of NS with **"improve"** and **"equal-improve"** statuses is that the PA operator will focus only NS with these two statuses which have possibility to give better solution. Next, total number of $NS(t,s)$ that have **"equal-improve"** status is divided by the sum of NS with **"improve"** and **"equal-improve"** statuses and multiplied by $1-\gamma$ to reduce the possibility of occurrence on the **"equal-improve"** status. In steps 6 and 7 (line 14 to 18 in Figure 5.3), if the random value (*PAR_rate*) that is generated earlier, is smaller than the value of PAR, then the program randomly executes the NS with **"improve"** status on the solution. Otherwise, the program randomly executes the NS with **"equal-improve"** status on the solution. In step 8 (line 20 to 22 in Figure 5.3), if the value of *HMCR* is greater than *Rand*, then the random consideration stage is executed. The program randomly executes the NS with **"non-improve"** status on the solution.

For the stopping criterion, if maximum improvisation (MI) is still not reached, then all the solutions are reset to **"improve"** status and the number of HMS is increased to the maximum improvisation. This condition indicates that the iterations are increased in terms of the HMS. The HM is also updated with the best solution, and the worst solution is removed. Finally, the CADENZA demonstrates the best solution in the HM.

5.4 Experimental Settings and Results

The experimental settings consist of two parts, the basic and adaptive parameters settings. There is common setting between the basic and adaptive which is the initial population that are produced using the enhanced KHE.

Based on a series of experiments using various combination of values, the basic HSA parameter for HMCR was set as 0.80. The PAR and RCR parameters were selected based on Equations 5.3 and 5.4 for HMC and RC operators, respectively. For example, if the random value of PAR_{rate} is more than 0.30 and less than 0.45, the $TasksMoveResource$ NS will be executed.

$$Adjust\ (s) = \begin{cases} NS_{\text{MeetsSwap}}\ if\ PAR_{rate} > 0.75\ AND\ PAR_{rate} \leq 1, \\ NS_{TasksSwap}\ if\ PAR_{rate} > 0.60\ AND\ PAR_{rate} \leq 0.75\ , \\ NS_{MeetsMoveTime}\ if\ PAR_{rate} > 0.45\ AND\ PAR_{rate} \leq 0.60\ , \\ NS_{TasksMoveResource}\ if\ PAR_{rate} > 0.30\ AND\ PAR_{rate} \leq 0.45, \\ NS_{BlockMeetsSwap}\ if\ PAR_{rate} > 0.15\ AND\ PAR_{rate} \leq 0.30, \\ NS_{KempeMeetMoveTime}\ if\ PAR_{rate} \leq 0.15 \end{cases} \quad (5.3)$$

$$Adjust\ (s) = \begin{cases} NS_{KempeMeetMove}, if\ RCR_{rate} > 0.75\ AND\ RCR_{rate} \leq 1, \\ NS_{TaskEjectingMoveResource}, if\ RCR_{rate} > 0.50\ AND\ RCR_{rate} \leq 0.75, \\ NS_{EjectingMeetMoveTime}\ , RCR_{rate} > 0.25\ AND\ RCR_{rate} \leq 0.50, \\ NS_{EjectingMeetMove}\ if\ RCR_{rate} \leq 0.25 \end{cases} \quad (5.4)$$

The adaptive HSA parameters, such as HMCR, RCR, and PAR, are set based on the proposed mathematical Equations 5.1 and 5.2. The value of α, β, and γ are set based on priority mechanism of HMCR, PAR, and RCR from previous literatures on fixed parameters HSA such as $\alpha = 0.8$, $\beta = 0.7$ and $\gamma = 0.7$. Finally, the number of iterations for the run is set with the number of HMS such as 8, 16, 32, 64, and 128 (based on the population of the initial solution produced for each dataset) multiplied by two to give double possibilities of each solution in population.

The result of DHSA is compared with basic HSA, meta-heuristics, and state-of-the-art methods of HSTP. The result values in Tables 5.1, 5.2, 5.3, and 5.4 indicate the objective function values that consist of the total number of hard and soft constraint violations. A value of 1.00447 denotes that the total number of hard constraint violations is 1, and the total penalty value of soft constraint violations is 447.

Table 5.1 demonstrates the comparison of result between basic HSA and DHSA. In comparative evaluation, the DHSA presented enhanced results across 21 datasets. However, the basic HSA proved advantageous on four datasets, due to the inherent randomness in NS. A parity in outcomes was observed in the remaining 14 datasets between the two algorithms. Thus, this comparison proves that DHSA is powerful compared to basic HSA. By using the average rank to measure performance, the DHSA recorded a score of 1.43. This score was lower than that of the basic HAS which is 1.89, thereby indicating that the DHSA outperforms the basic HSA in this comparison.

Table 5.2 shows the results of the DHSA compared with the best-known results using different metaheuristics methods since 2012, such as VNS (Fonseca et al., 2016), SA-ILS (Fonseca et al., 2016b), SA+ILS, GVNS, SGVNS (Teixeira et al., 2019), and HSA-SA (Shambour et al., 2013). When compared with existing metaheuristics methods, DHSA outperforms them on 15 instances (AU-SA-96, AU-TE-99, BR-SA-00, BR-SN-00, BrazilInstance5, BrazilInstance3, BrazilInstance7, VillageSchool, ArtificialSchool, Preveza2008, IT-I4-96, NL-KP-03, NL-KP-09, Kottenpark2008 and ZL-LW-09) and matches their results on 6 instances (SecondarySchool2, ElementarySchool, GR-H1-97, GR-P3-10, UniversityInstance5 and KS-PR-1) within the XHSTT data set.

Comparing DHSA with VNS (Fonseca et al., 2016), DHSA obtains 27 better and 3 equal solutions out of 39 instances. In a comparison with HSA-SA (Shambour et al., 2013) which has only produced results on 11 datasets, it is evidenced that DHSA obtains superior solutions in 6 instances and achieves parity in 2 instances, thereby underlining its comparative efficacy against HSA-SA. DHSA obtains 32 better solutions out of 39 instances compared with SA+ILS and GVNS (Teixeira et al., 2019),

and obtains 31 better solutions out of 39 instances compared with SGVNS (Teixeira et al., 2019). In a comparison against SA-ILS (Fonseca et al., 2016b) which has generated results for only 16 datasets, DHSA outperforms it in 8 datasets and produces equivalent results in 3 datasets. DHSA is ranked as the best approach as it has the lowest average value (1.92) compared to other approaches.

Table 5.3 illustrates the best results of HSTP obtained using different state-of-the-art methods i.e. mathematical-based methods which are A) Mixed-Integer linear Programming (Kristiansen et al., 2015a), B) Matheuristics (Fonseca et al., 2016), C) MaxSAT-based large neighbourhood search (Demirović & Musliu, 2017), and D) A Hidden Markov Model Approach to the Problem of Heuristic Selection (Kheiri & Keedwell, 2017). DHSA performs well in minimizing soft constraint violations, but it is not that competitive in minimizing hard constraint violations. In another words, DHSA is effective in minimizing soft constraint violations, but less competitive when it comes to minimizing hard constraint violations. Hard constraints represent non-negotiable conditions that must be satisfied for a solution to be feasible, making aforementioned methods minimization critical. While DHSA excels in reducing soft constraint violations, it lacks competitiveness in handling hard constraints, where the aforementioned methods offer superior robustness and effectiveness.

However, DHSA outperforms 5 instances (VillageSchool, SecondarySchool2, Preveza2008, Aigio2010 and UK-SP-06) and tied with 4 instances (ElementarySchool, GR-H1-97, GR-P3-10 and UniversityInstance5) compared with the state-of-the-art methods. In addition, the average rank is used to compare the performance of the four

algorithms (A, B, C, and D) with DHSA. DHSA ranked as the third place with 2.17 average value after B (Fonseca et al., 2016) and D (Kheiri & Keedwell, 2017).

Table 5.1 *Results of DHSA Compared to Basic HSA*

Instance	Basic HSA	DHSA
AU-BG-98	1.00557	1.00446
AU-SA-96	0.00006	0.00003
AU-TE-99	0.00087	0.00075
BR-SA-00	0.00011	0.00011
BR-SM-00	2.00125	2.00119
BR-SN-00	0.00096	0.00096
BrazilInstance5	0.00078	0.00068
BrazilInstance3	0.00062	0.00065
BrazilInstance7	0.00134	0.00134
BrazilInstance1	0.00045	0.00048
VillageSchool	0.00013	0.00013
DK-FG-12	0.01893	0.01832
DK-HG-12	12.04576	12.03419
DK-VG-09	2.04325	2.03327
ES-SS-08	0.00565	0.00542
FI-PB-98	0.00005	0.00005
FI-WP-06	0.00014	0.00013
FI-MP-06	0.00091	0.00091
SecondarySchool2	0.00002	0.00000
ElementarySchool	0.00003	0.00003
ArtificialSchool	3.00004	3.00004
GR-H1-97	0.00000	0.00000
GR-P3-10	0.00000	0.00000
GR-PA-08	0.00006	0.00005
UniversityInstance3	0.00006	0.00006
UniversityInstance5	0.00000	0.00000
Preveza2008	0.00002	0.00002
Aigio2010	0.00008	0.00004
IT-I4-96	0.00037	0.00038
KS-PR-11	0.00008	0.00004
NL-KP-03	0.00886	0.00806
NL-KP-05	2.01278	3.01143
NL-KP-09	7.04190	5.04190
Kottenpark2008	24.29205	14.32969
GEPRO	1.02006	1.00567
UK-SP-06	11.00908	11.00904
US-WS-09	0.00537	0.00126
ZL-LW-09	2.00034	0.00004
ZA-WD-09	2.00000	2.00000
Average Rank	**1.89**	**1.43**

As evidenced by the data above, Matheuristics (Fonseca et al., 2016), and A Hidden Markov Model Approach to the Problem of Heuristic Selection (Kheiri & Keedwell, 2017) emerge as the top methodologies in the literature review because of their consistently superior findings.

Table 5.2 *Results of DHSA Compared to Best-Known Metaheuristic Approaches*

Instance	DHSA	VNS	SA + ILS	SGVNS	GVNS	HSA-SA	SA-ILS
AU-BG-98	1.00446	2.00398	6.00450	**1.00401**	4.00370	-	-
AU-SA-96	**0.00003**	4.00021	14.00050	13.00046	12.00051	-	-
AU-TE-99	**0.00075**	1.00036	7.00161	7.00163	7.00151	-	-
BR-SA-00	**0.00011**	0.00022	-	-	-	0.00048	0.00032
BR-SM-00	2.00119	3.00099	0.00091	0.00090	**0.00094**	0.00162	1.00136
BR-SN-00	**0.00096**	0.00104	0.00149	0.00131	0.00148	0.00186	0.00160
BrazilInstance5	**0.00068**	-	0.00164	0.00149	0.00158	0.00148	-
BrazilInstance3	**0.00065**	-	-	-	-	0.00154	0.00101
BrazilInstance7	**0.00134**	-	0.00264	0.00248	0.00249	0.00234	-
BrazilInstance1	0.00048	-	0.00012	**0.00011**	**0.00011**	0.00020	-
VillageSchool	**0.00013**	-	-	-	-	-	-
DK-FG-12	0.01832	**0.01669**	-	-	-	-	-
DK-HG-12	12.03419	**12.0337**	-	-	-	-	-
DK-VG-09	2.03327	**2.02765**	-	-	-	-	-
ES-SS-08	0.00542	0.00415	-	-	-	-	**0.00012**
FI-PB-98	0.00005	**0.00000**	-	-	-	-	-
FI-WP-06	0.00013	0.00011	**0.00001**	**0.00001**	**0.00001**	-	-
FI-MP-06	0.00091	**0.00084**	0.00086	0.00088	0.00087	0.00090	-
SecondarySchool2	**0.00000**	-	-	-	-	**0.00000**	**0.00000**
ElementarySchool	**0.00003**	-	-	-	-	**0.00003**	**0.00003**
ArtificialSchool	**3.00004**	-	-	-	-	-	-
GR-H1-97	**0.00000**	**0.00000**	-	-	-	-	-
GR-P3-10	**0.00000**	**0.00000**	-	-	-	-	-
GR-PA-08	0.00005	**0.00003**	-	-	-	-	0.00008
UniversityInstance3	0.00006	-	-	-	-	-	**0.00005**
UniversityInstance5	**0.00000**	-	-	-	-	-	**0.00000**
Preveza2008	**0.00002**	-	-	-	-	-	-
Aigio2010	0.00004	-	-	-	-	-	0.00000
IT-I4-96	**0.00038**	0.00042	-	-	-	0.00082	0.00061
KS-PR-11	**0.00004**	**0.00004**	-	-	-	-	-
NL-KP-03	**0.00806**	0.01151	0.01409	0.01281	0.01216	-	0.05355
NL-KP-05	3.01143	8.04460	0.01078	**0.00877**	0.00881	-	24.13930
NL-KP-09	**5.04190**	8.08370	-	-	-	-	19.05565
Kottenpark2008	**14.32969**	-	-	-	-	-	-
GEPRO	1.00567	-	1.00566	1.00441	**1.00434**	-	-
UK-SP-06	11.00904	53.01524	0.18092	**0.12466**	0.12542	-	-

US-WS-09	0.00126	**0.00124**	-	-	-	-	-
ZL-LW-09	**0.00004**	0.00016	0.00022	0.00024	0.00024	-	-
ZA-WD-09	2.00000	3.00000	-	-	-	-	**0 .00441**
Average Values	**1.92**	**2.12**	**3.53**	**2.53**	**2.8**	**3.18**	**2.68**
Rank	**1**	**2**	**7**	**3**	**5**	**6**	**4**

Note: *Dash (-)* denotes that there are no results provided by the author.

Table 5.3 *Results Obtained Using DHSA Compared to State-of-the-art Methods*

Instance	DHSA	A	B	C	D
AU-BG-98	1.00446	3.00494	2.00398	-	**0.00520**
AU-SA-96	0.00003	8.00052	3.00021	-	**0.00002**
AU-TE-99	0.00075	1.00140	1.00036	-	**0.00061**
BR-SA-00	0.00011	**0.00005**	0.00005	0.00057	0.00010
BR-SM-00	2.00119	0.00061	**0.00052**	0.00214	2.00117
BR-SN-00	0.00096	0.00060	**0.00035**	0.00352	0.00101
BrazilInstance5	0.00068	**0.00030**	-	0.00224	-
BrazilInstance3	0.00065	**0.00026**	-	0.00075	-
BrazilInstance7	0.00134	**0.00122**	-	0.00603	-
BrazilInstance1	0.00048	-	-	-	-
VillageSchool	**0.00013**	-	-	-	-
DK-FG-12	0.01832	2.23705	0.01668	-	**0.01522**
DK-HG-12	12.03419	293.3211	12.03371	-	**12.0263**
DK-VG-09	2.03327	20.18966	2.02765	-	**2.02731**
ES-SS-08	0.00542	0.00357	**0.00351**	-	0.00517
FI-PB-98	0.00005	-	**0.00000**	0.01309	0.00008
FI-WP-06	0.00013	**0.00001**	0.00002	0.00812	0.00007
FI-MP-06	0.00091	0.00088	**0.00077**	0.00504	0.00089
SecondarySchool2	**0.00000**	-	-	0.03523	-
ElementarySchool	0.00003	-	-	0.00003	-
ArtificialSchool	3.00004	-	-	**0.00012**	-
GR-H1-97	**0.00000**	-	**0.00000**	**0.00000**	**0.00000**
GR-P3-10	**0.00000**	-	**0.00000**	0.02329	**0.00000**
GR-PA-08	0.00005	0.00008	**0.00003**	0.00141	0.00004
UniversityInstance3	0.00006	**0.00005**	-	0.00007	-
UniversityInstance5	**0.00000**	-	-	**0.00000**	-
Preveza2008	**0.00002**	-	-	0.05617	-
Aigio2010	**0.00004**	-	-	0.04582	-
IT-I4-96	0.00038	**0.00027**	**0.00027**	0.16979	0.00038
KS-PR-11	0.00004	0.00003	**0.00000**	0.29946	0.00003
NL-KP-03	0.00806	0.01410	0.01103	-	**0.00466**
NL-KP-05	3.01143	0.01078	8.04460	-	**0.00811**
NL-KP-09	5.04190	**0.09035**	7.64470	-	2.07495
Kottenpark2008	14.32969	-	-	-	-
GEPRO	1.00567	**1.00566**	-	-	-
UK-SP-06	**11.00900**	-	53.01524	-	19.01294
US-WS-09	0.00126	-	**0.00124**	-	0.00512
ZL-LW-09	0.00004	-	**0.00000**	0.01039	0.00052
ZA-WD-09	2.00000	**0.00000**	**0.00000**	**0.00000**	9.00000
Average Values	**2.17**	**2.25**	**1.8**	**2.95**	**2.08**

Rank	**3**	**4**	**1**	**5**	**2**

Note: *Dash (-)* denotes that there are no results provided by the author.

Table 5.4 presents a comparison of three state-of-the-art methods: DHSA, Matheuristics (Fonseca et al., 2016) (B), and A Hidden Markov Model Approach to the Problem of Heuristic Selection (Kheiri & Keedwell, 2017) with average results from different random seeds.

However, Matheuristic method contains some results that produced from the best results (denoted by an asterisk (*)) rather than an average results. In column 2, DHSA have 12 of rank 1, 8 of rank 2 and 3 of rank 3 which contributed to 1.57 average value that ranked as the first place compared to the other two methods. The second place goes to Matheuristic, and the third place goes to A Hidden Markov Model Approach to the Problem of Heuristic Selection.

Table 5.4 *Comparative Performance of DHSA, A Hidden Markov Model Approach to the Problem of Heuristic Selection, and Matheuristic Methods with Different Random Seeds*

Instances	**DHSA**	**A Hidden Markov Model Approach to the Problem of Heuristic Selection**	**Matheuristic**
AU-BG-98	**1, 451.2** (1)	5, 542 (3)	2, 398* (2)
AU-SA-96	**0, 2.8** (1)	2, 15 (2)	3, 21* (3)
AU-TE-99	**0, 70.2** (1)	2, 89 (3)	1, 36* (2)
BR-SA-00	0, 10.6 (2)	0, 35 (3)	**0, 5.8** (1)
BR-SM-00	2, 119 (2)	6, 114 (3)	**0, 63.6** (1)
BR-SN-00	0, 93.4 (2)	0, 129 (3)	**0, 51.6** (1)
DK-FG-12	0, 1828 (3)	0, 1806 (2)	**0, 1668*** (1)
DK-HG-12	12, 3457.4 (3)	**12, 2863** (1)	12, 3371* (2)
DK-VG-09	2, 3260.2 (3)	2, 2827 (2)	**2, 2765*** (1)
ES-SS-08	0, 543 (2)	3, 49 (3)	**0, 474.8** (1)
FI-WP-06	0, 13 (2)	0, 27 (3)	**0, 2*** (1)

GR-P3-10	**0, 0** (1)	1, 29 (2)	**0, 0*** (1)
Preveza2008	**0, 2** (1)	-	-
IT-I4-96	0, 37.2 (2)	0, 62 (3)	**0, 32.2** (1)
KS-PR-11	**0, 2.2** (1)	0, 18 (3)	0, 9 (2)
NL-KP-03	**0, 847** (1)	0, 933 (2)	0, 1257.8 (3)
NL-KP-05	**2.6, 1193.8** (1)	4, 1161 (2)	14.4, 5677.8 (3)
NL-KP-09	**5, 14085** (1)	12, 8500 (3)	11.8, 11545 (2)
Kottenpark2008	**13.4, 32999.4** (1)	-	15, 68015.2 (2)
GEPRO	**1, 524** (1)	-	-
UK-SP-06	**10.8, 912.4** (1)	12, 14061 (2)	53, 1524* (3)
ZL-LW-09	0, 10 (2)	2, 57 (3)	**0, 0*** (1)
ZA-WD-09	2, 0 (2)	14, 0 (3)	**0, 57.8** (1)
Average Values	**1.57**	**2.55**	**1.67**
Rank	**1**	**3**	**2**

Note: The number can be comprehended as (hard average, soft average (rank)), where an asterisk (*) denotes indicates the utilization of the best solution rather than an average, as no average calculation is employed in their method

5.5 Analysis and Discussion

This section presents the analysis and discussion on the performance of DHSA regarding its level gaps on parameters tuning and multi-run. Table 5.5 shows the decreased level gap results between DHSA and MCHR (initial solutions method that described in Chapter Four). The column with ↓Gap level soft and ↓Gap level hard demonstrates the decreased level of gap for soft and hard constraints, which is calculated by using Equation 3.1. The dash (-) sign at the Lower Bound column indicates that the instances do not have the lower bound value. The double equal (==) sign at ↓GAP Level Soft and Hard shows that the solution results is the same when compared with the initial solution. DHSA can improve the solution for 22 out of 39 data instances compared to MCHR.

It is also found that the DHSA can minimize the soft constraint violations with huge to slight different. The soft constraint violation for US-WS-09 is minimized with huge

improvement, that is from 537 to 126 with 0.0573 of decreased level gap and is close to the lower bound solution. The AU-TE-99 is slightly minimized with 0.8209 of decreased level gap of soft constraint violation. Kottenpark2008 and ZL-LW-09 are minimized with 0.8750 and 0 of decreased level gap of hard constraint violation, respectively. In addition, both data instances also are minimized with 0.9127 and 0.11765 of decreased level gap of soft constraint violation, respectively. There are 14 data instances that have not improved for both soft and hard constraint violations.

Figure 5.6 shows the performance of DHSA that represent by value of gap level decreased and DHSA rates (HMCR, RCR, PAR1 and PAR2) against the iteration position and solution number on certain datasets: 1) small datasets (BrazilInstance5 and ZA-LW-09), 2) middle datasets (Aigio2010, AU-BG-98, AU-TE-99 and ES-SS-08) and 3) large datasets (NL-KP-09 and DK-VG-09). The PAR1, PAR2 and RCR represent the range of improve, equal-improve or non-improve behavioural status, respectively. In Figure 5.6 (a), there is a sudden rise of PAR2 from value zero (iteration 2) to value one (iteration 3), indicating that the NS in the iteration contains with highly 'improve' (from 0 to 0.68) and low 'equal-improve' (from 0.68 to 1) status which enhanced the BrazilInstance5 dataset due to the influence of the NS behavioural status on past iteration. In this case, the RCR has a low chance to execute compared to HMCR due to the influence of linkage parameters. In Figure 5.6 (b), both PAR2 and RCR show a sharp increase, reaching a value of one.

Table 5.5 *DHSA Compared with MCHR and Its Decreased Level Gap Results*

Instance	Lower bound	MCHR (Initial Solution)	DHSA	↓GAP Level Soft	↓GAP Level Hard
AU-BG-98	-	1.00557	1.00446	0.8007	==

AU-SA-96	0.00000	0.00006	0.00003	0.5000	==
AU-TE-99	0.00020	0.00087	0.00075	0.8209	==
BR-SA-00	0.00005	0.00011	0.00011	==	==
BR-SM-00	0.00051	2.00125	2.00119	0.9189	==
BR-SN-00	0.00035	0.00096	0.00096	==	==
BrazilInstance5	0.00019	0.00078	0.00068	0.8305	==
BrazilInstance3	0.00024	0.00065	0.00065	==	==
BrazilInstance7	0.00040	0.00134	0.00134	==	==
BrazilInstance1	0.00041	0.00048	0.00048	==	==
VillageSchool	0.00010	0.00013	0.00013	==	==
DK-FG-12	0.00412	0.01901	0.01832	0.9537	==
DK-HG-12	-	12.04670	12.03419	0.7321	==
DK-VG-09	2.00000	2.04344	2.03327	0.7659	==
ES-SS-08	-	0.00565	0.00542	0.9593	==
FI-PB-98	-	0.00005	0.00005	==	==
FI-WP-06	-	0.00014	0.00013	0.9286	==
FI-MP-06	0.00077	0.00091	0.00091	==	==
SecondarySchool2	-	0.00002	0.00000	0	==
ElementarySchool	0.00003	0.00003	0.00003	==	==
ArtificialSchool	-	3.00004	3.00004	==	==
GR-H1-97	-	0.00000	0.00000	==	==
GR-P3-10	-	0.00000	0.00000	==	==
GR-PA-08	0.00003	0.00006	0.00005	0.6667	==
UniversityInstance3	0.00005	0.00006	0.00006	==	==
UniversityInstance5	-	0.00000	0.00000	==	==
Preveza2008	-	0.00002	0.00002	==	==
Aigio2010	-	0.00008	0.00004	0.5000	==
IT-I4-96	0.00027	0.00040	0.00038	0.8462	==
KS-PR-11	-	0.00008	0.00004	0.5000	==
NL-KP-03	-	0.00808	0.00806	0.9975	==
NL-KP-05	0.00089	3.01143	3.01143	==	==
NL-KP-09	-	7.04190	5.04190	==	0.7142
Kottenpark2008	0.02795	16.35854	14.32969	0.9127	0.8750
GEPRO	1.00000	1.01606	1.00567	0.3308	==
UK-SP-06	-	11.00908	11.00904	1.0000	==
US-WS-09	0.00101	0.00537	0.00126	0.0573	==
ZL-LW-09	-	2.00034	0.00004	0.11765	0
ZA-WD-09	-	2.00000	2.00000	==	==

This suggests that the execution of NS is either have equal-improve or non-improve statuses for the ZL-LW-09 dataset. The PAR2 value hits an extreme point at the third iteration, and the RCR value reaches an exceptionally high point at the fourth iteration. This is also the case for AU-BG-98 data instance, as demonstrated in Figure 5.6 (d). As depicted in Figure 5.6 (c), only lines for decreased gap level (↓Gap), PAR1 and PAR2

are shown due to that the HMCR and RCR values match the values of PAR1 and PAR2, which are 0.8 and 1, respectively. This perhaps because of the total of NS equal-improve and improve statuses matching the total number of NS non-improve status. This is also similar case for AU-TE-99 and ES-SS-08 data instances, as demonstrated in Figure 5.6 (e) and Figure 5.6 (f), respectively. The RCR line is not depicted in either of the graphs because the RCR value aligns with the PAR2 value. This might be due to the total of NS equal-improve and improve statuses being equal to the overall count of NS non-improve statuses. In Figure 5.6 (g), for the NL-KP-09 data instance, two distinct decreased gap levels exist, based on hard and soft constraints. The adaptive parameters effectively reduce the decreased hard gap level, however, unable to reduce the decreased soft gap level. In addition, between iterations 3 and 4 smooth decline of the decreased hard gap level can be observed, as a result of the increasing of PAR2 and decreasing of RCR, which aids the execution of NS that produce equal-improve status to function with a higher probability. In Figure 5.6 (h) for DK-VG-09 data instance, there is a gradual decrease trend in the gap level during all iteration. From iterations 6 to 12, all parameters show stable values. However, from iterations 13 to 18, the values of all parameters illustrate unstable movement which increased or decreased at certain iterations. This is due to the producing of NS improve, equal-improve, or non-improve statuses for each iteration.

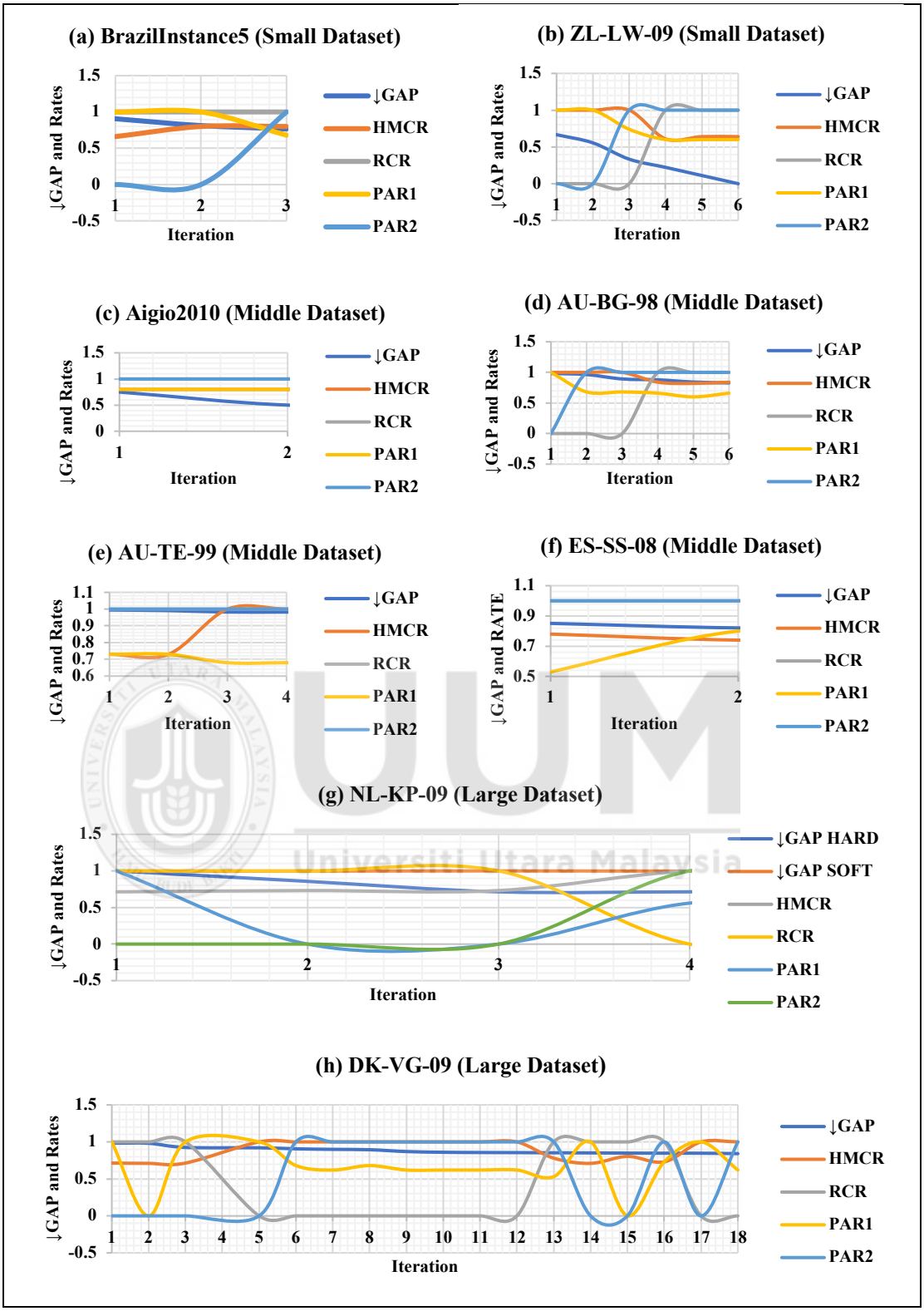


Figure 5.6. The decreased gap level constraints and adaptive parameters

Significant improvements in terms of gap levels are observed more notably in larger datasets compared to smaller and medium sizes datasets. The adaptive parameters have a substantial impact on the NL-KP-09 and DK-VG-09 data instances.

The multi-run of DHSA was tested on (a) small dataset, (b) medium dataset, and (c) large dataset for 20 iterations. Figure 5.7 shows the results graph for the multi-runs on the soft and hard constraints level gap values. The level gap value of soft constraints for small datasets presented in Figure 5.7 (a) shows decreasing trend in the second run for 2 datasets (BrazilInstance5 and FI-WP-06) and in the third run to fourth run for ES-SS-08 dataset. However, the level gap value of soft constraints for ZA-LW-09 dataset was not improved. This perhaps is due to that the soft constraints for small datasets are extremely close to the current optimal or lower bound values. Level gap value of hard constraints for ZL-LW-09 data instance reaches to zero after fourth runs of DHSA. The soft constraints' level gap value for medium datasets, as depicted in Figure 5.7 (b), indicates a decline in the first run for the IT-I4-96 dataset, until sixth run for the AU-BG-98 dataset, and an improvement to half the level gap value until the second run for the Aigio2010 dataset. This likely suggests that these medium dataset's soft constraints are nearing the current optimal or lower limit values. Figure 5.7 (c) presents the level gap of the soft and hard constraints for large datasets. For NL-KP-09 dataset, the level gap of the soft constraints was not improved possibly due to an unavailability of suitable NS. However, the level gap of the hard constraints of NL-KP-09 dataset was decreased until second run. The level gap of the soft constraints for the DK-VG-09 dataset is depicted with decrement until seventeenth run. The level gap of the soft constraints for the US-WS-09 dataset was enormously improved approximately 50% at first run and still decreased until nineteenth run.

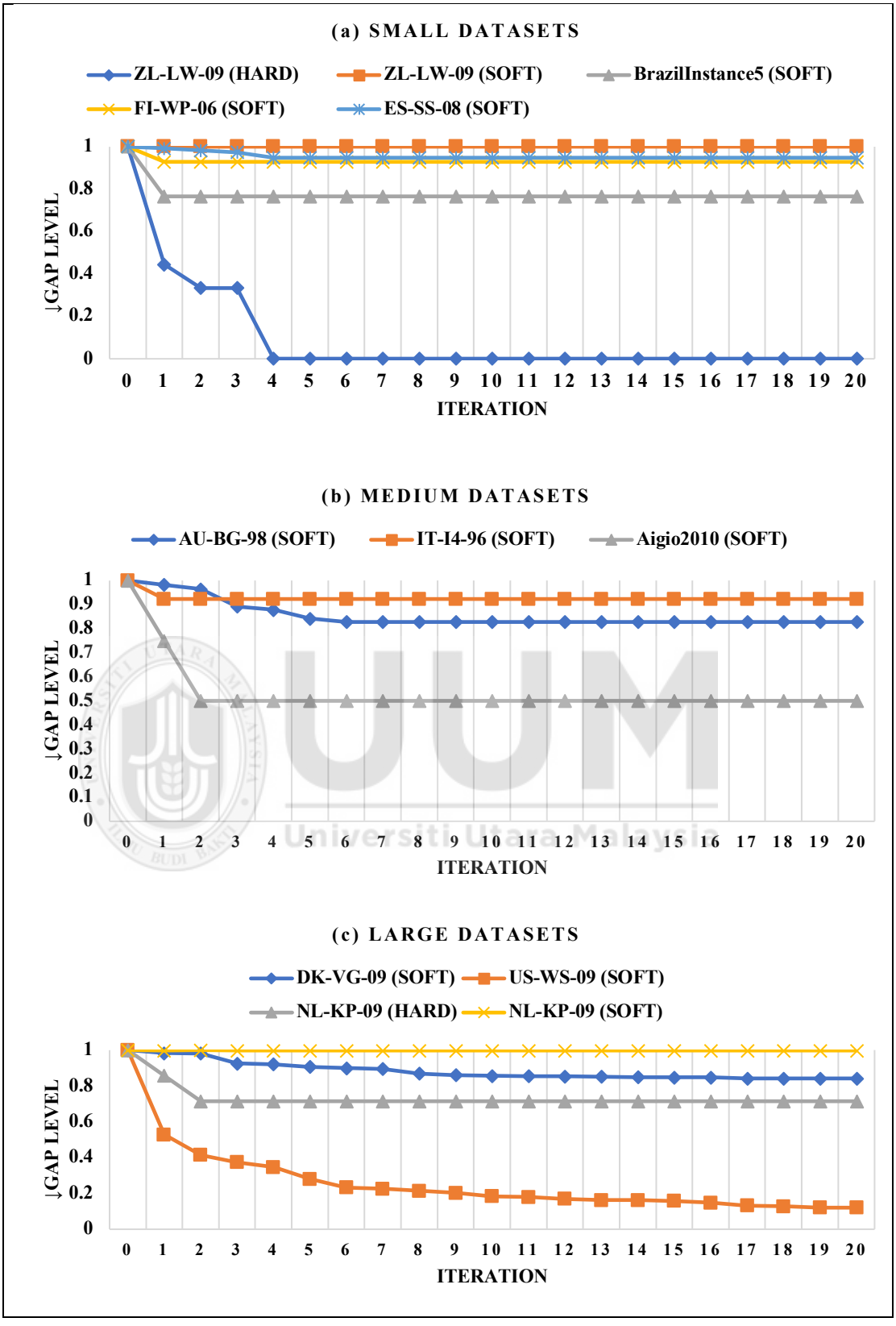


Figure 5.7. The decreased gap level constraints

5.5.1 Comparative Analysis of DHSA with Existing Approaches

DHSA represents a significant evolution in optimizing search exploration and exploitation balance. This is achieved through innovative strategies, notably the restriction of parameter changes throughout the search process and the incorporation of the solution's behavioral state into parameter tuning. Additionally, it bridges the gap between adaptive and static parameters previously associated with the domain, offering a more cohesive approach to parameter adjustment. The radar chart (Figure 5.8) and bar chart (Figure 5.9) visually compare various adaptive HSAs across six key criteria: Scalability, Robustness, Efficiency, Generalizability, Innovation, and Applicability. The algorithms compared include those proposed by Mahdavi et al. (2007), Omran & Mahdavi (2008), Wang & Huang (2010), Chen et al. (2012), Peraza et al. (2016), Wang et al. (2019), Quan et al. (2021), Hasanipanah et al. (2022) and the proposed DHSA. In Figure 5.9, each axis represents one of the criteria, with values ranging from 1 to 5. The proposed DHSA demonstrates superior performance across all criteria, achieving the maximum score of 5 in each category, as calculated in Section 3.5.1 using a matrix-based evaluation framework. This widely recognized method, rooted in established practices in computer science, engineering, and software metrics evaluation, systematically assesses performance across scalability (Cormen et al., 2022), robustness (Rousseeuw & Leroy, 2005), efficiency (Bishop, 2006), generalizability (Jordan & Mitchell, 2015), innovation (He et al., 2016), and applicability (Kitchin, 2014). This approach highlights DHSA's conceptual strengths and operational capabilities without requiring additional experiments, underscoring its exceptional scalability, robustness, efficiency, generalizability, innovation, and applicability in solving complex optimization problems.

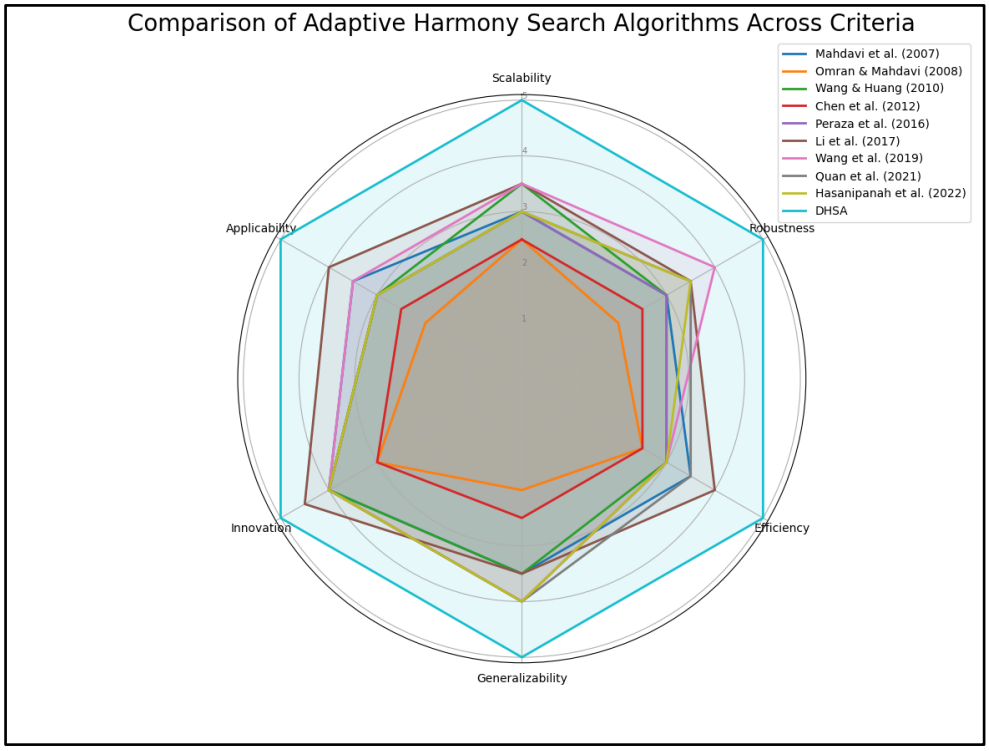


Figure 5.8 Comparison of adaptive harmony search algorithms across criteria

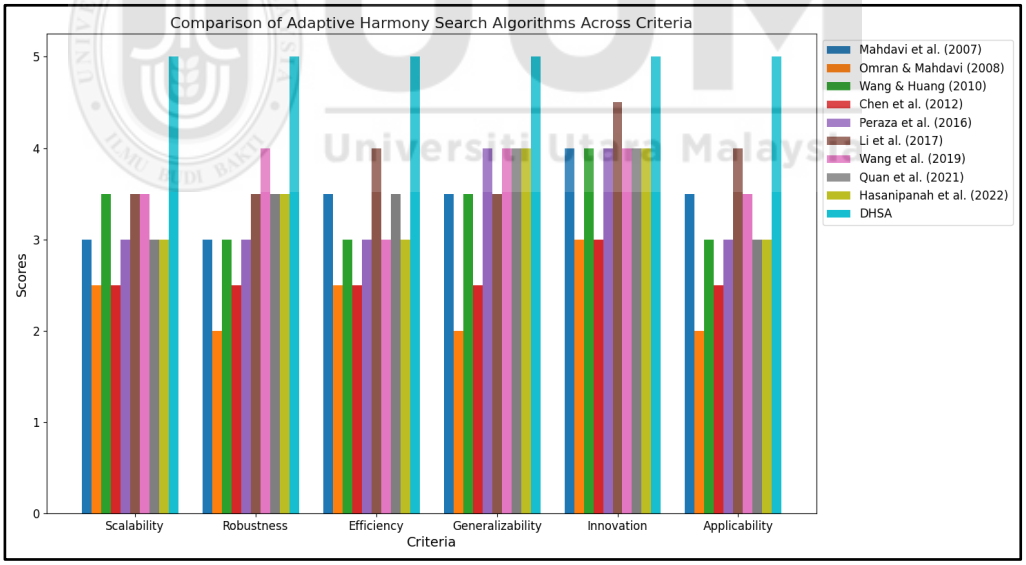


Figure 5.9 Bar chart comparison of adaptive harmony search algorithms across criteria

Table 5.6 briefly shows comparative analysis of the proposed DHSA for HSTP against existing adaptive HSA approaches on scalability, robustness, efficiency, generalizability, innovation, and applicability in solving complex optimization problems.

Table 5.6 *Criteria Comparative Analysis of DHSA for HSTP Against Existing Adaptive HSA Approaches*

Criteria / Approaches	Scalability	Robustness	Efficiency	Generalizability	Innovation	Applicability
(Mahdavi et al., 2007)	Score: 3 (Good scalability on tested benchmarks, but not necessarily representative of real-world problems)	Score: 3 (High robustness on simpler benchmarks, needs more real-world testing)	Score: 3.5 (More efficient than standard HS, but evaluated on simpler benchmarks)	Score: 3.5 (Good generalizability within simpler benchmarks)	Score: 4 (Innovative approach that improved performance)	Score: 3.5 (Effective on standard benchmarks, needs validation on real-world applications)
(Omran & Mahdavi, 2008)	Score: 2.5 (Lower scalability due to reliance on simpler benchmarks)	Score: 2 (Lower robustness due to simpler benchmark testing)	Score: 2.5 (Lower efficiency as evaluated on simpler problems)	Score: 2 (Lower generalizability due to simpler benchmarks)	Score: 3 (Innovative but needs validation on complex problems)	Score: 2 (Lower applicability for complex, real-world problems)
(Wang & Huang, 2010)	Score: 3.5 (Good scalability on tested benchmarks, but not necessarily representative of real-world problems)	Score: 3 (High robustness on simpler benchmarks, needs more real-world testing)	Score: 3 (More efficient than standard HS, but evaluated on simpler benchmarks)	Score: 3.5 (Good generalizability within simpler benchmarks)	Score: 4 (Innovative approach that improved performance)	Score: 3 (Effective on standard benchmarks, needs validation on real-world applications)
(Chen et al., 2012)	Score: 2.5 (Lower scalability due to simpler benchmarks, with performance drops on larger instances)	Score: 2.5 (Moderate robustness with some variability in results)	Score: 2.5 (Improved efficiency but still limited on simpler benchmarks)	Score: 2.5 (Some generalizability within simpler benchmarks)	Score: 3 (Innovative approach with dynamic control parameters)	Score: 2.5 (Effective on standard benchmarks, needs validation on real-world applications)
(Peraza et al., 2016)	Score: 3 (Good scalability on tested benchmarks, but not necessarily representative of real-world problems)	Score: 3 (High robustness on simpler benchmarks, needs more real-world testing)	Score: 3 (More efficient than standard HS, but evaluated on simpler benchmarks)	Score: 4 (Good generalizability within simpler benchmarks)	Score: 4 (Innovative approach with fuzzy logic for dynamic parameter adaptation)	Score: 3 (Effective on standard benchmarks, needs validation on real-world applications)

(Li et al., 2017)	Score: 3.5 (Good scalability on complex problems, but may not fully represent real-world scalability)	Score: 3.5 (High robustness on tested problems, but more real-world testing needed)	Score: 4 (More efficient than standard HS, but evaluated on specific complex problems)	Score: 3.5 (Good generalizability within the scope of assembly sequence planning)	Score: 4.5 (Innovative approach with dynamic parameter control and opposition-based learning)	Score: 4 (Effective on complex assembly sequence planning problems, needs validation on broader real-world applications)
(Wang et al., 2019)	Score: 3.5 (Good scalability on tested benchmarks, but not necessarily representative of real-world problems)	Score: 3 (High robustness on simpler benchmarks, needs more real-world testing)	Score: 3 (More efficient than standard HS, but evaluated on simpler benchmarks)	Score: 4 (Good generalizability within simpler benchmarks)	Score: 4 (Innovative approach with differential mutation strategies and novel pitch adjustment rules)	Score: 3.5 (Effective on standard benchmarks, needs validation on real-world applications)
(Quan et al., 2021)	Score: 3 (Good scalability on specific complex problems, but may not fully represent broader real-world scalability)	Score: 3.5 (High robustness on specific complex problems, but more real-world testing needed)	Score: 3.5 (More efficient than standard HS, but evaluated on specific complex problems)	Score: 4 (Good generalizability within the scope of dynamic path planning)	Score: 4 (Innovative approach with self-adaptive mechanisms and Morphin Algorithm)	Score: 3 (Effective on specific complex problems, needs validation on broader real-world applications)
(Hasanipanah et al., 2022)	Score: 3 (Good scalability on specific complex problems, but may not fully represent broader real-world scalability)	Score: 3.5 (High robustness on specific complex problems, but more real-world testing needed)	Score: 3 (More efficient than standard HS, but evaluated on specific complex problems)	Score: 4 (Good generalizability within the scope of flyrock prediction problems)	Score: 4 (Innovative approach with ANN-adaptive mechanisms and dynamic parameter control)	Score: 3 (Effective on specific complex problems, needs validation on broader real-world applications)
DHSA	Score: 5 (Excellent scalability, maintaining performance on large and complex problems)	Score: 5 (High robustness with consistent performance across diverse problem instances)	Score: 5 (Highly efficient, with fewer function evaluations and better performance on complex problems)	Score: 5 (Superior generalizability across diverse, complex problems)	Score: 5 (Highly innovative with comprehensive parameter adjustments and adaptive strategies)	Score: 5 (Highly effective in real-world applications, especially complex NP-complete problems)

Scalability: Mahdavi et al. (2007) exhibited good scalability (3.0) in complex problems, showcasing their ability to handle intricate scenarios effectively. Wang and Huang (2010) also demonstrated good scalability (3.5), highlighting their improved performance on high-dimensional problems. Omran and Mahdavi (2008) showed moderate scalability (2.5) on high-dimensional problems, but their approach was primarily validated on simpler benchmarks. Chen et al. (2012) and Peraza et al. (2016) demonstrated decent scalability (2.5 and 3.0, respectively) but encountered challenges with larger instances. Li et al. (2017) exhibited good scalability (3.5) and improved performance on complex problems, emphasizing their adaptability. Wang et al. (2019) demonstrated improved scalability (3.5) in high-dimensional problems, showcasing their capability to handle theoretical or simulation contexts. Hasanipanah et al. (2022) and Quan et al. (2021) both demonstrated good scalability (3.0 each) within their specialized domains, indicating their robustness in specific applications. In comparison, DHSA excels in scalability with a score of 5.0 by dynamically adjusting parameters based on both iteration position and the total number of solutions. This adaptive, multi-dimensional strategy **ensures thorough exploration** of the search space and **efficient exploitation of promising regions**, maintaining optimal performance as problem size and complexity increase. DHSA's ability to handle large, complex NP-complete problems, such as high school timetabling, underscores its superior scalability and adaptability in real-world, dynamic environments.

Robustness: Most algorithms demonstrated high robustness with low variability in performance across different runs and problem instances. Mahdavi et al. (2007) showed a robustness score of 3.0, indicating consistent performance. Wang and Huang (2010) also achieved a robustness score of 3.0, maintaining low standard deviations in their

results. Peraza et al. (2016) similarly showed robustness with a score of 3.0. Li et al. (2017) had a slightly higher robustness score of 3.5, reflecting their stable performance across various complex problems. Wang et al. (2019) demonstrated high robustness with a score of 4.0, showcasing low standard deviations in results. Hasanipanah et al. (2022) also achieved a robustness score of 3.5, supported by low root mean square error (RMSE) values. Quan et al. (2021) maintained a robustness score of 3.5, indicating stable performance in dynamic path planning scenarios. Omran and Mahdavi (2008) exhibited moderate robustness with a score of 2.0, reflecting some variability in simpler benchmark tests. In comparison, DHSA excels in robustness with a perfect score of 5.0. It leverages historical neighborhood move data to influence future iterations, enhancing the diversity within the search process and ensuring the algorithm avoids getting stuck in local optima. This method ensures that DHSA can consistently find high-quality solutions across multiple runs, **effectively balancing exploration and exploitation**. By maintaining and enhancing diversity within the search process, DHSA achieves lower standard deviations, demonstrating its superior robustness in various optimization challenges.

Efficiency: Chen et al. (2012) and Mahdavi et al. (2007) required significant computation time and function evaluations, indicating moderate efficiency with scores of 2.5 and 3.5, respectively. This shows that while they performed well, they demanded substantial computational resources. Wang and Huang (2010), with a score of 3.0, and Peraza et al. (2016), with a score of 3.0, improved efficiency by requiring fewer function evaluations for convergence. Li et al. (2017) achieved a higher efficiency score of 4.0, indicating significant improvements in computational resource management through dynamic parameter control. Wang et al. (2019) also demonstrated high

efficiency with a score of 3.0, showcasing fewer function evaluations compared to traditional HS algorithms. Hasanipanah et al. (2022) and Quan et al. (2021) both demonstrated high efficiency, scoring 3.0 and 3.5, respectively, reflecting their ability to achieve faster convergence and fewer function evaluations in their specialized domains. Omran and Mahdavi (2008) achieved better performance with fewer function evaluations, reflected in an efficiency score of 2.5. In comparison, DHSA stands out with a perfect efficiency score of 5.0. By adaptively tuning parameters based on the objective function's behavioral state, DHSA requires fewer function evaluations and achieves faster convergence. This adaptive mechanism ensures that the algorithm efficiently navigates the search space, **exploiting** known good regions while continuing to explore new areas. Thus, DHSA maintains a **balance between exploration and exploitation**, resulting in highly efficient performance across diverse optimization challenges.

Generalizability: Chen et al. (2012) and Mahdavi et al. (2007) performed well on multiple optimization problems but required parameter adjustments, indicating moderate generalizability with scores of 2.5 and 3.5, respectively. This reflects their ability to solve various problems but with some need for manual tuning. Wang and Huang (2010), Peraza et al. (2016), Li et al. (2017), and Wang et al. (2019) all showed good generalizability across various benchmark functions, with scores of 3.5, 4.0, 3.5, and 4.0 respectively, highlighting their effectiveness in different contexts with minimal adjustments. Hasanipanah et al. (2022) and Quan et al. (2021) demonstrated effective solutions to specialized problems, earning generalizability scores of 4.0 each, showing their robustness in particular domains. Omran and Mahdavi (2008) exhibited strong generalizability across various benchmark functions, with a score of 2.0, indicating

their ability to handle different types of problems but primarily within simpler benchmarks. In comparison, DHSA excels in generalizability with a perfect score of 5.0. DHSA effectively solves different instances of NP-complete problems like high school timetabling without significant parameter adjustments, thanks to its linkage mechanism between adaptive and static parameters. This linkage mechanism allows DHSA to adapt to a wide range of optimization problems, ensuring high versatility across diverse domains. This capability enables DHSA to **balance exploration and exploitation** effectively by adapting to the specific characteristics of each problem, making it a highly generalizable algorithm..

Innovation: Chen et al. (2012) introduced hybridization with Genetic Algorithms (GA), earning a moderate innovation score of 3.0 for integrating additional optimization techniques to enhance performance. Mahdavi et al. (2007) dynamically adjusted the Pitch Adjusting Rate (PAR) and bandwidth (bw), resulting in an innovation score of 4.0, reflecting significant improvements to the standard Harmony Search algorithm. Wang and Huang (2010) used a self-adaptive mechanism, achieving an innovation score of 4.0, highlighting their novel approach to parameter adaptation. Peraza et al. (2016) utilized fuzzy logic for dynamic parameter adaptation, earning an innovation score of 4.0 for incorporating advanced techniques to improve algorithm adaptability. Li et al. (2017) implemented dynamic parameter control and opposition-based learning, resulting in a high innovation score of 4.5, showcasing their advanced methods for enhancing search efficiency. Wang et al. (2019) incorporated Differential Evolution (DE) mutation strategies, earning an innovation score of 4.0 for integrating robust optimization techniques. Hasanipanah et al. (2022) used dynamic parameter control based on harmony memory and Artificial Neural Networks (ANN), earning an

innovation score of 4.0, highlighting their advanced approach to enhancing algorithm performance. Quan et al. (2021) introduced neighbors and optimal learning strategies, achieving an innovation score of 4.0 for their novel enhancements to the Harmony Search algorithm. Omran and Mahdavi (2008) applied global-best concepts from swarm intelligence, resulting in an innovation score of 3.0, reflecting their integration of swarm intelligence principles. In comparison, DHSA excels in innovation with a perfect score of 5.0. DHSA integrates multiple innovative strategies, including comprehensive parameter adjustment (incorporating RCR and iteration size) and the adaptive use of historical behavioral data. These innovations enhance the algorithm's ability to balance exploration and exploitation by dynamically adjusting search strategies based on real-time feedback from the search process, significantly improving performance and solution quality across various optimization challenges.

Applicability: Chen et al. (2012) demonstrated effectiveness in multiple optimization problems, achieving a moderate applicability score of 2.5. Mahdavi et al. (2007) effectively solved real-world engineering problems, earning a higher applicability score of 3.5, while Wang and Huang (2010) also showed strong applicability in real-world contexts, with a score of 3.0. Peraza et al. (2016) and Li et al. (2017) both demonstrated significant applicability in engineering and industrial problems, earning scores of 3.0 and 4.0, respectively, reflecting their robust application in practical scenarios. Wang et al. (2019) effectively addressed high-dimensional function optimization problems, earning an applicability score of 3.5. Hasanipanah et al. (2022) focused on flyrock prediction in surface mining, showcasing their algorithm's practical utility with a score of 3.0. Quan et al. (2021) solved dynamic path planning problems, demonstrating strong applicability with a score of 3.0. Omran and Mahdavi (2008) effectively solved both

continuous and discrete optimization problems, earning a moderate applicability score of 2.0. In comparison, DHSA excels in applicability with a perfect score of 5.0. It solves practical scheduling problems, such as high school timetabling, with robust convergence and high effectiveness. DHSA's ability to maintain a balance between exploring new solutions and exploiting known good solutions ensures it can effectively address complex, real-world optimization problems with high accuracy and reliability. This superior applicability in real-world, NP-complete problems highlights DHSA's robustness and versatility, making it a highly effective tool for solving diverse and intricate optimization challenges.

DHSA achieves a superior balance between exploration and exploitation through its adaptive approach. By continuously adjusting parameters based on the search space's current state and historical data, DHSA ensures that it thoroughly explores the search space while efficiently exploiting promising areas to find optimal solutions. This balance is crucial for solving complex, real-world problems effectively, making DHSA a highly robust and adaptable optimization algorithm.

In summary, DHSA satisfies the exploration and exploitation balance across all criteria, demonstrating exceptional scalability, robustness, efficiency, generalizability, innovation, and applicability. This makes DHSA a powerful tool for tackling a wide range of complex optimization challenges, particularly those that are NP-complete.

5.5.2 Comparative Analysis of the Dynamic Balancing Value Between Exploration and Exploitation in DHSA and Existing Approaches

The following analysis compares the dynamic balancing value between exploration and exploitation for DHSA against several existing modified HSA approaches (Omran & Mahdavi, 2008), (Wang & Huang, 2010), (Wang et al., 2019) and (Misni & Lee, 2021). The comparison is conducted over 50, 100, and 200 iterations, as illustrated in the respective graphs (Figures 5.10, 5.11 and 5.12). In the respective figures, the maximum exploration value is zero and the maximum exploitation value is one. Initially, the dynamic balancing value was set to zero to demonstrate how DHSA and the modified HSA approaches explore and exploit the search space.

In the initial 50 iterations as shown in Figure 5.10, the DHSA demonstrates a steady increase in the balance value from approximately 0.1 to around 0.3. This indicates a gradual shift from exploration to exploitation. In contrast, the existing approaches show minimal change in their balance values. Omran & Mahdavi (2008) maintain a constant low balance value, emphasizing stable exploration. Wang & Huang (2010) also show negligible variation, keeping the balance value close to the initial value. Similarly, Wang et al. (2019) and Misni & Lee (2021) approaches remain almost constant, highlighting a consistent exploration phase without a significant shift towards exploitation.

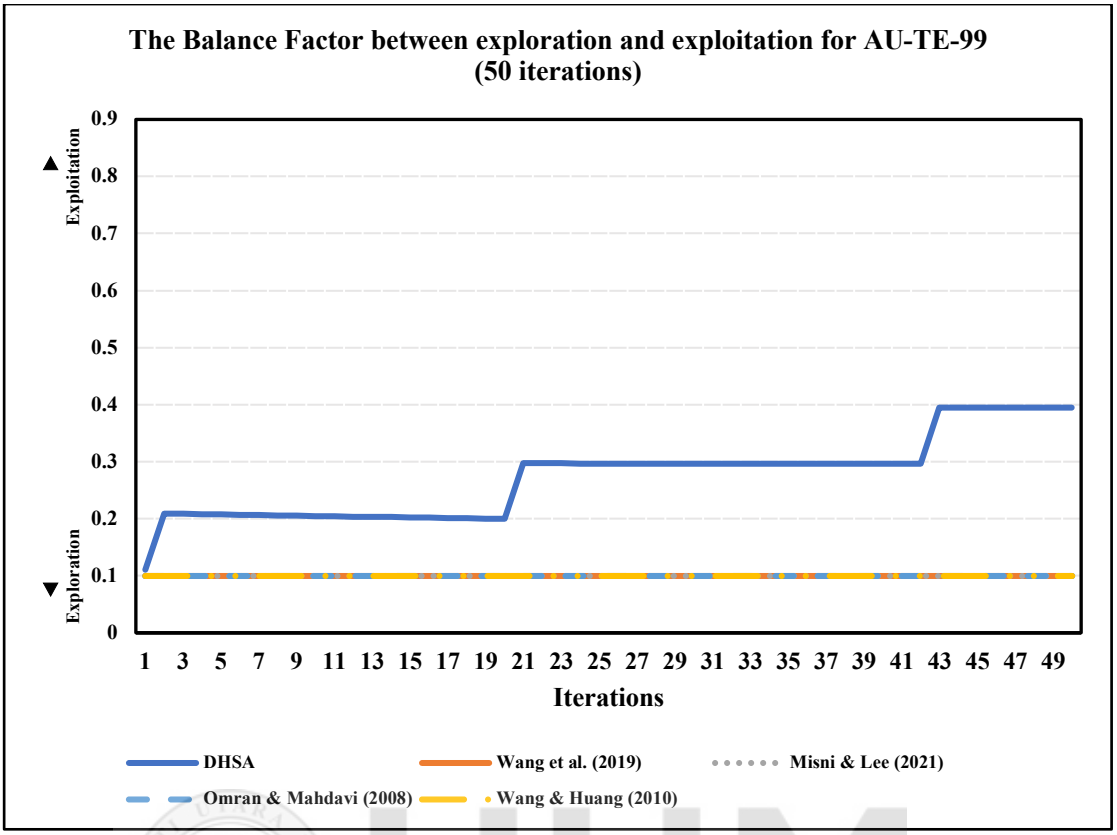


Figure 5.10 The Balance Factor between Exploration and Exploitation for AU-TE-99 (50 iterations)

Over 100 iterations as shown in Figure 5.11, the DHSA exhibits a more pronounced increase in the balance value, reaching approximately 0.6. This significant rise indicates a stronger emphasis on exploitation as the iterations progress, suggesting that the DHSA adapts dynamically to optimize the solution space more effectively. The existing approaches, however, continue to show minimal changes. Omran & Mahdavi (2008) and Wang & Huang (2010) maintain their initial low balance values, indicating a persistent focus on exploration. Wang et al. (2019) and Misni & Lee (2021) also display negligible variation, reflecting a consistent strategy with limited adaptation towards exploitation.

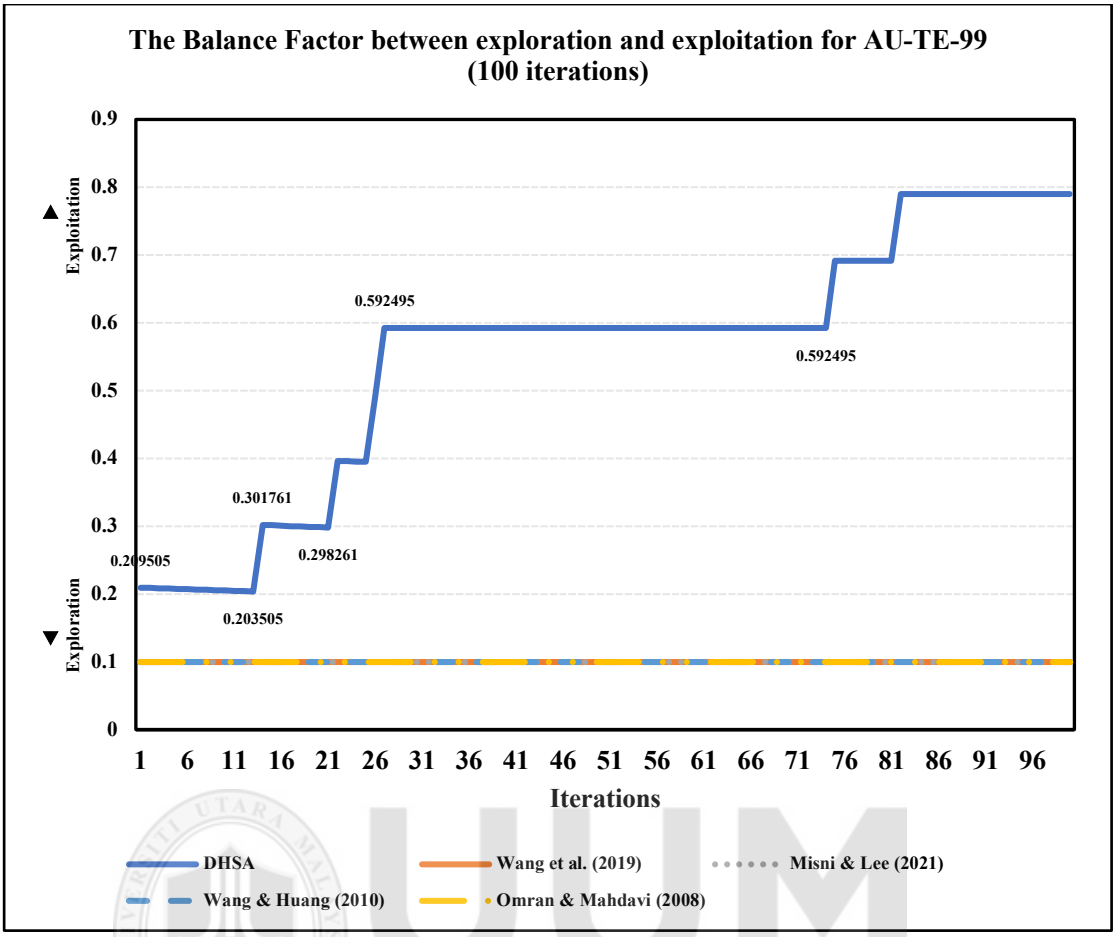


Figure 5.11 The Balance Factor between Exploration and Exploitation for AU-TE-99 (100 iterations)

At 200 iterations as shown in Figure 5.12, the DHSA's balance value reaches around 0.7, demonstrating a robust shift towards exploitation. This dynamic adjustment enables the algorithm to refine and optimize the best-found solutions effectively. In comparison, the existing approaches remain largely unchanged. Omran & Mahdavi (2008), Wang & Huang (2010), and Wang et al. (2019) continue to show minimal fluctuations in their balance values, maintaining a steady exploration phase. Misni & Lee (2021) also exhibits a constant balance value, underscoring a static exploration strategy.

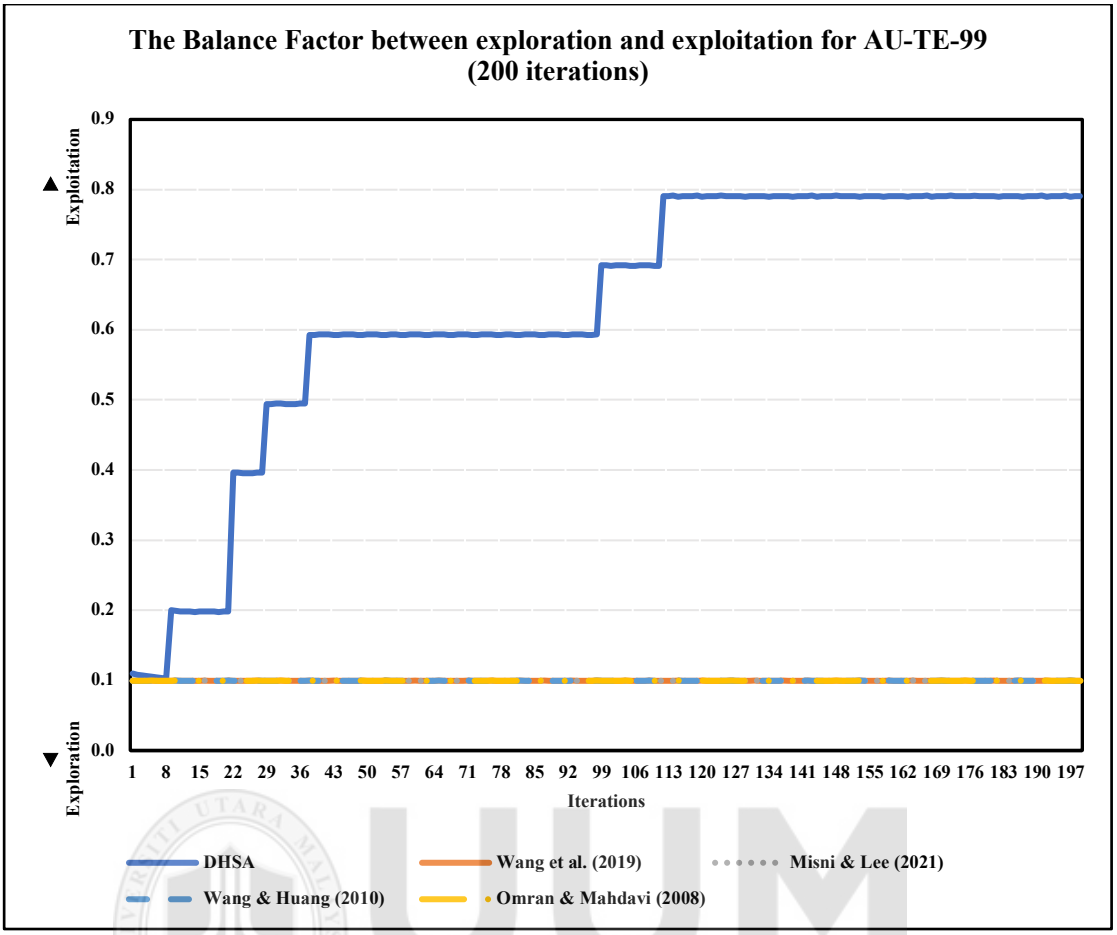


Figure 5.12 The Balance Factor between Exploration and Exploitation for AU-TE-99 (200 iterations)

The comparative analysis reveals that the DHSA's dynamic balance value adapts significantly over the iterations, effectively transitioning from exploration to exploitation. This adaptability contrasts with the existing approaches, which maintain relatively constant balance values, emphasizing exploration with limited exploitation. The DHSA's ability to dynamically adjust the balance value enhances its optimization efficiency, leading to improved performance in navigating and optimizing the solution space.

5.6 Summary

This study proposes the DHSA that use adaptive parameters and several NSs homogeneously to improve the HSTP based on (1) iteration position and solution number, (2) behavioral status, and (3) parameters linkages. The DHSA delivers significantly better results due to the nature of the adaptive parameters and improves the majority of instances compared with other best-known solutions proposed in the literature. Thus, the establishment of DHSA is feasible. Consequently, the DHSA rapidly enhances the solution search through exploration and exploitation at any starting point from harmonies/populations. The adaptive parameters assist the DHSA to automatically identify which neighborhood move should be utilized for the subsequent iterations. The adaptive parameter values of HMCR, RCR, and PACR are recalculated at each iteration, thereby supporting the DHSA to provide high priority of neighborhood moves for improving the solution search. This study demonstrates the superiority of DHSA with adaptive parameters compared with nonadaptive parameters of HSA when dealing with a complex HSTP. The selected NS in the DHSA, aggressively drives the solution towards minimizing the objective function. This is particularly true for small and medium datasets, which results in the inability to reduce the objective function further. In addition, the constraints of the solution with a group of events must be addressed. This condition requires a consideration for an improvement to DHSA to explore more on the local solution space. Chapter Six describes the hybridization of DHSA with great deluge algorithm (GDA) which is one of the local search methods in metaheuristic to further exploit the local solution space and thus improve the solution.

CHAPTER SIX

A HYBRID ADAPTIVE HARMONY SEARCH AND GREAT DELUGE ALGORITHM FOR SCHOOL TIMETABLING

6.1 Introduction

This chapter describes the hybridization of DHSA with great deluge algorithm (GDA) to further improve the solution of HSTP from previous Chapter Five. Firstly, Section 6.2 explains the implementation of basic GDA to solve HSTP. However, the nature of HSTP that contains two types of constraints were solved as one constraint by the basic GDA. Therefore, Section 6.3 highlights the modification of GDA which split the process of solving the constraints to soft and hard constraints components. Even though the modified GDA can produce better solution compared to basic GDA, only one solution is produced at a time and the NS is selected based on randomness of all NS moves. Section 6.4 that describes the hybridization of DHSA with modified GDA, to produce the population of solutions and randomness of specific NS moves based on adaptive parameters of DHSA. The experiments and results are presented in Section 6.5, while the discussion is highlighted in Section 6.6. Finally, the last part contains the chapter's summary.

6.2 Basic Great Deluge Algorithm (BGDA)

This section describes the implementation process of basic Great Deluge algorithm (BGDA) for solving HSTP. The steps of this method are detailed in the flowchart in Figure 6.1. The first step is setting the inputs for BGDA such as creation of an initial

solution using enhanced KHE algorithm, setting the maximum number of iterations (totalIter), the rain speed (rainSpeed) and 10 NS which similar to NS that used in basic HSA as shown in Section 5.2. The totalIter and rainSpeed parameters are set based on preliminary experimentation (as explained in 3.4.1 Section). Step 2 involves the calculation of the water level (current objective function), and decay rate, which is shown in lines 1 and 2 of Figure 6.2. The calculation of decay rate is based on literature review as given in Equation 2.9.

Finally, Step 3 in Figure 6.1 is about the iterative phase of BGDA, which is illustrated in lines 3 to 12 in Figure 6.2. This stage is all about fine-tuning a single solution. NS is selected randomly to be used in modifying the current solution, as shown in line 4 in Figure 6.2. BGDA accepts solutions that are better or similar to the current solution. In addition, BGDA also accepts solutions that are better or similar to the water level, as shown in lines 5-8 in Figure 6.2. Otherwise, the new solutios are not accepted as shown in lines 9 and 10 in Figure 6.2. This flexibility helps BGDA to avoid local optima and keep searching for the best overall solution. During the search, the water level is continuously decreased by subtracting it with the decay rate as shown in line 11 in Figure 6.2. The implementation of BGDA for solving HSTP involves only one type of constraint which is combined both soft and hard constraints and does not accommodate the two types of constraints (soft and hard) of HSTP simultaneously. This leads to a lack of control over the decreasing water level. Next section describes the modification of BGDA to deal with this issue.

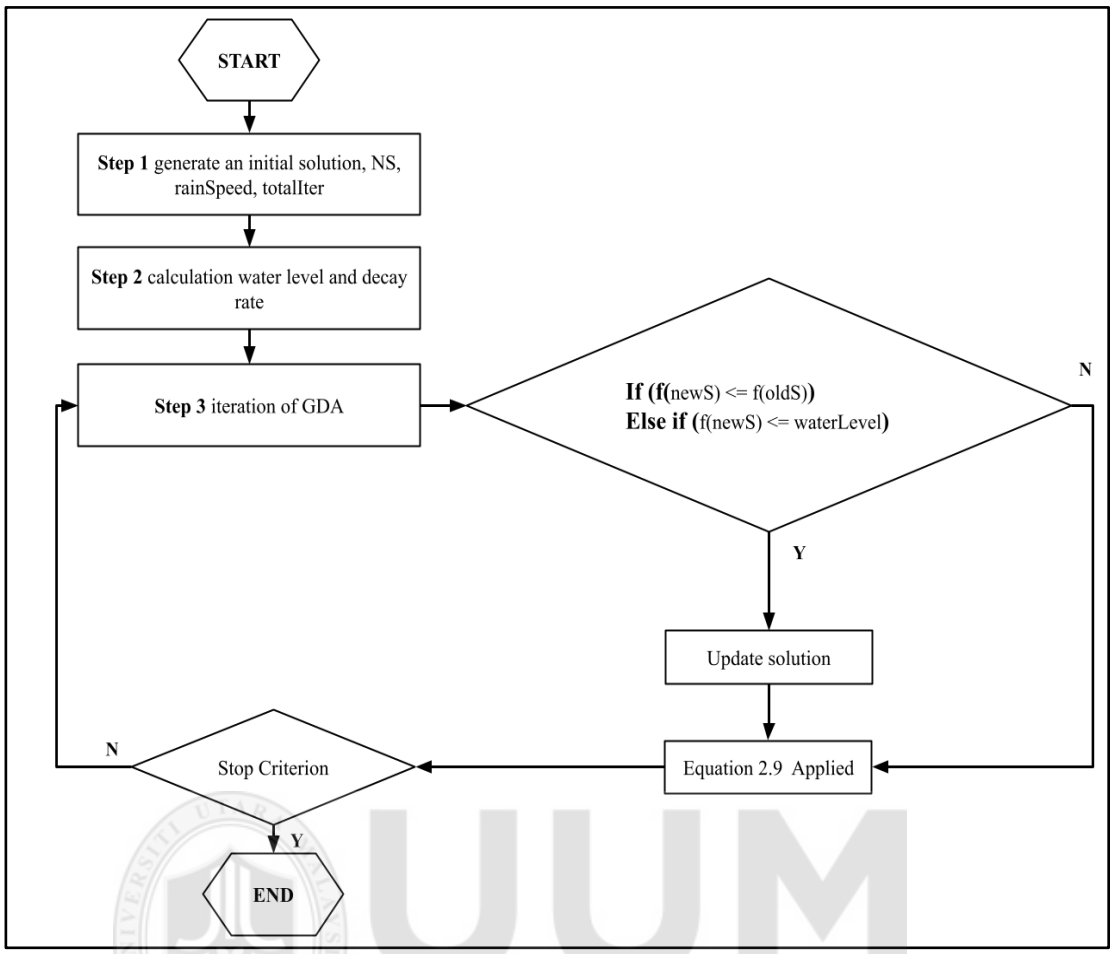


Figure 6.1. BGDA flowchart

Algorithm 6.1 Basic Great Deluge Algorithm (MGSA)

Input : $soln_{old}$, NS, $rainSpeed$, $totalIter$
Output: $soln_{new}$

$waterLevel \leftarrow f(soln_{old})$
$decayRate \leftarrow f(soln_{old}) \times rainSpeed \div totalIter$
for $i \leftarrow 0$ **to** $totalIter$ **do**
  $soln_{new} \leftarrow generateNeighbour(soln_{old}, NS)$
  if $f(soln_{new}) \leq f(soln_{old})$ **then**
    $soln_{old} \leftarrow soln_{new}$
  else if $f(soln_{new}) \leq waterLevel$ **then**
    $soln_{old} \leftarrow soln_{new}$
  else
    Not Accepted
  end if
  $waterLevel \leftarrow waterLevel - decayRate$
end for
return $Soln_{new}$

Figure 6.2. Pseudocode for BGDA

6.3 Modified Great Deluge Algorithm (MGDA)

This section illustrates the modification of BGDA in the setting of water level and decay rate. The BGDA treats both soft and hard constraints of HSTP as a single constraint, leading to insufficient control over the water level adjustment process. This approach fails to adequately address the different types of constraints, necessitating an enhancement that separates the handling of soft and hard constraints to improve the solution quality and ensure better diversification in the search process. The existing enhanced GDA methods (Burke et al., 2006; Landa-Silva & Obit, 2008a; Mushi, 2011) lack non-linearity in the water level acceptance with the cost function, which limits diversification in the search process. Therefore, the modified great deluge algorithm (MGDA) will have two types of water levels which represents the water level of soft constraints ($waterLevelSoft$) and water level of hard constraints ($waterLevelHard$). In addition, MGDA also have two decay rates for soft ($decayRateSoft$) and hard $decayRateHard$ constraints. The flowchart shown in Figure 6.3 explains how the MGDA works. The initial process involves configuring inputs for MGDA, which includes creating an initial solution via an enhanced KHE algorithm namely as MCHR (presented in Chapter Four), determining the maximum iteration count (totalIter), setting the rain speed (rainSpeed) and 10 NS that similar to the NS employed in the BGDA. The totalIter and rainSpeed values are chosen based on initial experimental results. In Step 2, the water level soft and hard are set using the soft and hard objective function of HSTP, respectively. Moreover, both of soft and hard decay rates are calculated using the Equations 6.1 and 6.2, respectively (as shown in line 1 to 4 in Figure 6.4). In Step 3, NS is selected randomly to be used in modifying the current solution (as shown in line 5 in Figure 6.4). MGDA accepts solutions that are better or similar to the current solution (as shown in line 6 in Figure 6.4). Moreover, MGDA will

accept a new solution if its soft and hard costs are less than or equal to the *waterLevelSoft* and *waterLevelHard* parameters, respectively. Alternatively, if the soft and hard costs of the new solution are the same as the current solution, it will also be accepted. Otherwise, the new solutios are not accepted (as shown in line 7 to 13 in Figure 6.4). During the search, both of the *waterLevelSoft* and *waterLevelHard* parameters are continuously decreased by subtracting it with $decayRateSoft$ and $decayRateHard$, as shown in Equations 6.3 and 6.4, respectively (as shown in lines 14 and 15 in Figure 6.4).

$$decayRateSoft = f(Soln_{cur})_{Soft} \times \frac{rainSpeed}{totalIter} \tag{6.1}$$

$$decayRateHard = f(Soln_{cur})_{Hard} \times \frac{rainSpeed}{totalIter} \tag{6.2}$$

$$waterLevelSoft = waterLevelSoft - decayRateSoft \tag{6.3}$$

$$waterLevelHard = waterLevelHard - decayRateHard \tag{6.4}$$

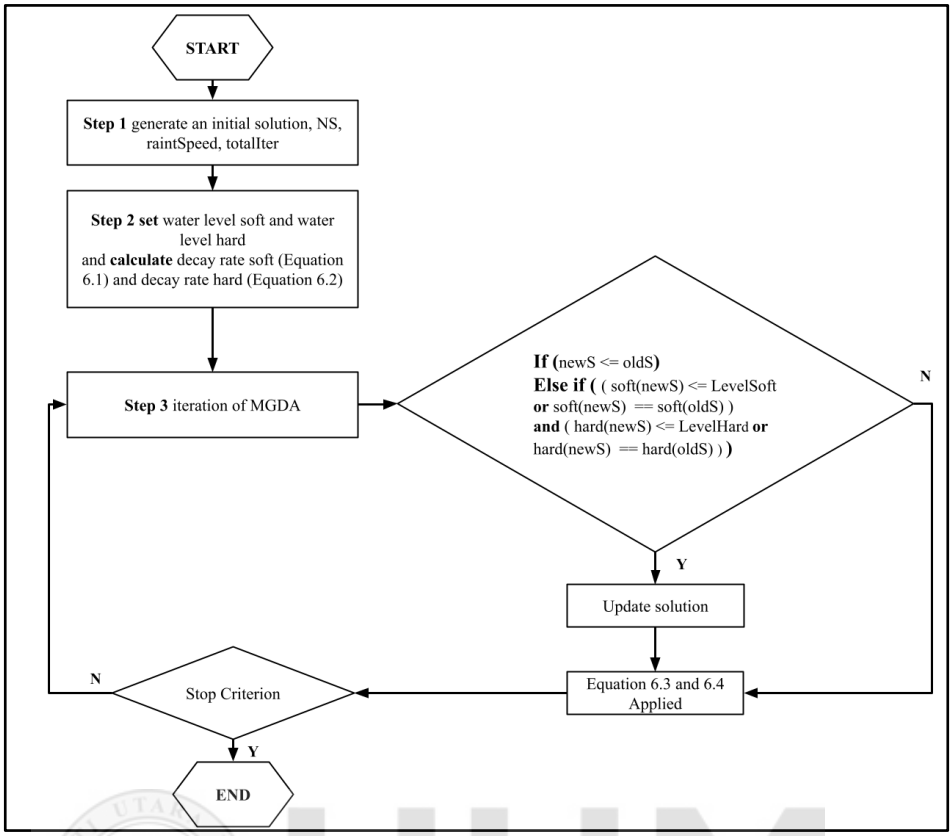


Figure 6.3 Modified GDA Flowchart

Algorithm 6.2 Modified Great Deluge Algorithm (MGDA)

Input : $soln_{old}$, NS, $rainSpeed$, $totalIter$
Output: $soln_{new}$

$waterLevelSoft \leftarrow f(soln_{old})_{soft}$
$waterLevelHard \leftarrow f(soln_{old})_{hard}$
$decayRateSoft \leftarrow f(soln_{old}) \times rainSpeed \div totalIter$
$decayRateHard \leftarrow f(soln_{old}) \times rainSpeed \div totalIter$
for $i \leftarrow 0$ **to** $totalIter$ **do**
  $soln_{new} \leftarrow generateNeighbour(soln_{old}, NS)$
  if $f(soln_{new}) \leq f(soln_{old})$ **then**
    $soln_{old} \leftarrow soln_{new}$
  else if $\left(f(soln_{new})_{soft} \leq waterLevelSoft \quad or \quad f(soln_{new})_{soft} = f(soln_{old})_{soft}\right) \quad AND$
  $\left(f(soln_{new})_{hard} \leq waterLevelHard \quad or \quad f(soln_{new})_{hard} = f(soln_{old})_{hard}\right)$ **then**
    $soln_{old} \leftarrow soln_{new}$
  else
    Not Accepted
  end if
  $waterLevelSoft \leftarrow waterLevelSoft - decayRateSoft$
  $waterLevelHard \leftarrow waterLevelHard - decayRateHard$
end for
return $Soln_{new}$

Figure 6.4. Pseudocode for MGDA

6.4 Hybridization of DHSA with MGDA

In the DHSA, the chosen NS actively pushes the solution towards reducing the objective function. This is especially the case for smaller and medium-sized datasets, which often results in a limit to how much the objective function can be decreased. Additionally, the selection of NS in MGDA is managed by the randomness of all possible NS moves.

Algorithm 6.3 Hybridization DHSA with MGSA
Input: HM, NS, MI
Output: HM

```
While t < MI Do
    Rand ← rand(0,1)
    RCR ← Equation 5.4
    HMCR ← 1 − RCR
    /* Harmony Memory Consideration (HMC)*/
    if HMCR ≤ Rand then
        /*Pitch Adjustment (PA)*/
        if SizeOf(equal_improve) + SizeOf(great_improve)
                > SizeOf(improve) then
            PAR2 ← Equation 6.1
            PAR1 ← 1 − PAR2
        else
            PAR1 ← Equation 6.1
            PAR2 ← 1 − PAR1
        End if
        PAR_rate ← rand(0,1)
        if PAR_rate ≤ PAR1 AND PAR_rate > PAR2 then
            Algorithm 6.2 Applied (rand (NS_i ⇒ NS_i = improve))
        else if PAR_rate ≤ PAR2 then
            Algorithm 6.2 Applied (rand (NS_i ⇒ NS_i = equal_improve))
        else
            Algorithm 6.2 Applied (rand (NS_i ⇒ NS_i = great_improve))
        End if
    Else
        /* Random Consideration (RC)*/
        Algorithm 6.2 Applied (rand (NS_i ⇒ NS_i = non_improve))
    End if
End while
return HM
```

Figure 6.5. Pseudocode for hybridization DHSA with MGDA

The hybridization of DHSA with MGDA will overcome these issues by incorporating MGDA water level into the pitch adjustment operator in DHSA that considers the behavioral status of neighborhood movements during the search space of MGDA. Within the DHSA, the threshold accepting water level in the MGDA is used to characterize each solution as "great-improve" which is signaling a significant improvement in the search for optimal solutions. Furthermore, there will be some slight changes made to the PAR calculations in DHSA.

Figure 6.5 shows the pseudocode of hybridization of DHSA with MGDA. The pseudocode inherited similar pseudocode of DHSA in Chapter Five. However, the differences are presented from lines 8 to 26. In harmony memory consideration (HMC) operator, for lines 8 through 14, if the sum of NS statuses "equal-improve" and "great-improve" exceeds the total number of "improve" statuses, the PAR2 value will be determined using Equation 6.1. Following this, the PAR1 value is calculated by subtracting 1 from the PAR2 value. If this is not the case, the PAR1 value is determined using Equation 6.1, and the PAR2 value is then calculated by subtracting the PAR1 value from 1. In Equation 6.1, the γ value for **"improve"** status is set with high priority compared to the **"equal-improve"** and **"great-improve"** statuses. Both **"equal-improve"** and **"great-improve"** statuses have the same equal priority. The reason is to guide the search space to produce better solutions during the search space. For lines 15 to 26, firstly, a random value (between 0 to 1) is assigned to PAR_{rate}. If the PAR_{rate} value is equal or less than PAR1, and greater than PAR2, NS with **"improve"** status will be randomly executed. Moreover, if the PAR_{rate} value is equal or less than PAR2, NS with **"equal-improve"** status will be randomly implemented. Otherwise, NS with **"great-improve"** status will be randomly applied. For random consideration (RC)

operator (which executed if the HMC operator is not executed), NS with **"non-improve"** status will be randomly applied. Each of **NS with "improve", "equal-improve", "great-improve"** and **"non-improve"** statuses will be randomly implemented using Algorithm 6.2 as shown in Figure 6.4.

$$PAR(t,s) = \left(\left(\left(\frac{(1-\gamma)}{2}\right) \times \frac{\sum_{EI=1}^{n} NS_{EI}(t,s)}{\{\sum_{EI=1}^{n} NS_{EI}(t,s) + \sum_{Im=1}^{n} NS_{Im}(t,s) + \sum_{GI=1}^{n} NS_{IG}(t,s)\}}\right) + \left(\left(\frac{(1-\gamma)}{2}\right) \times \frac{\sum_{GI=1}^{n} NS_{GI}(t,s)}{\{\sum_{EI=1}^{n} NS_{EI}(t,s) + \sum_{Im=1}^{n} NS_{Im}(t,s) + \sum_{GI=1}^{n} NS_{GI}(t,s)\}}\right)\right) + \left(\gamma \times \frac{\sum_{Im=1}^{n} NS_{Im}(t,s)}{\{\sum_{EI=1}^{n} NS_{EI}(t,s) + \sum_{Im=1}^{n} NS_{Im}(t,s) + \sum_{GI=1}^{n} NS_{IG}(t,s)\}}\right) \quad (6.1)$$

Where:

- ***t*:** it is the value of iteration factor.
- ***s*:** it is the value of solution position.
- ***Im*:** It is neighbourhood move position that gives better solution in MGDA.
- ***EI*:** It is neighbourhood move position that gives equal solution in MGDA.
- ***GI***: It is neighbourhood move position that gives threshold acceptance solution in MGDA
- ***n*:** It is the size of NSs.
- ***γ*:** It is the contribution value based on the previous mechanism of setting for PAR and neighbourhood moves.

6.5 Experimental Setting and Results of BGDA, MGDA and Hybrid DHSA with MGDA

This section explains the experimental setup as well as the results, and discussion about comparisons of BGDA, MGDA and hybrid DHSA with MGDA.

Tables 6.1 and 6.2 show parameter settings of BGDA and MGDA. The *rainSpeed* was set as 0.4 and 0.2 for BGDA and MGDA, respectively. The number of iterations (*totalIter*) used for both BGDA and MGDA in this study is 200 multiply the size of data instance. The initial water level for BGDA was set using objective function value. The initial soft and hard water levels for MGDA were set using soft and hard objective function values, respectively. All the setting values were determined after a series of experiments employing various parameter value combinations. Table 6.3 shows the results of both BGDA and MGDA that compared with the initial solution.

Table 6.1

The BGDA Parameter Settings

Parameters	Values
rainSpeed	0.4
totalIter	200 × size of data instance
waterLevel	Total objective function value

Table 6.2

The MGDA Parameter Settings

Parameters	Values
rainSpeed	0.2
totalIter	200 × size of data instance
waterLevelSoft	Soft objective function value
waterLevelHard	Hard objective function value

By comparing with initial solutions, the BGDA produce better solutions for 16 out of 39 data instances. However, for large datasets (NL-KP-0, NL-KP-05, NL-KP-09, UK-SP-06, DK-FG-12) and medium dataset (FI-PB-98) the BGDA produce worse solutions. MGDA performed respectable results in 22 out of 39 datasets when compared with the initial solutions.

Table 6.3

Results of the MGDA, BGDA, and the Initial Solution

Instance	Initial solution	**BGDA**	**MGDA**
Aigio2010	0.00008	**0.00006**	**0.00006**
ArtificialSchool	3.00006	3.00006	3.00006
AU-BG-98	1.00557	**1.00469**	**1.00469**
AU-SA-96	0.00005	0.00005	0.00005
AU-TE-99	0.00087	**0.00085**	**0.00085**
BrazilInstance1	0.00048	0.00048	0.00048
BrazilInstance3	0.00065	0.00065	0.00065
BrazilInstance5	0.00083	**0.00068**	**0.00068**
BrazilInstance7	0.00130	0.00130	0.00130
BR-SA-00	0.00011	0.00011	0.00011
BR-SM-00	2.00125	**2.00119**	**2.00119**
BR-SN-00	0.00096	0.00096	0.00096
DK-FG-12	0.01926	0.02096	**0.01907**
DK-HG-12	12.04790	**12.03521**	12.03664
DK-VG-09	2.04382	**2.03434**	2.03584
ElementarySchool	0.00003	0.00003	0.00003
ES-SS-08	0.00565	**0.00555**	**0.00555**
FI-MP-06	0.00091	0.00091	0.00091
FI-PB-98	0.00005	0.00007	**0.00005**
FI-WP-06	0.00014	**0.00013**	**0.00013**
GEPRO	1.01735	1.00705	**1.00613**
GR-H1-97	0.00000	0.00000	0.00000
GR-P3-10	0.00000	0.00000	0.00000
GR-PA-08	0.00006	**0.00005**	**0.00005**
IT-I4-96	0.00042	**0.00040**	**0.00040**
Kottenpark2008	19.35165	14.42141	**14.34855**
KS-PR-11	0.00009	0.00006	**0.00004**
NL-KP-03	0.00929	0.01252	**0.00921**
NL-KP-05	1.02226	3.01682	**1.01850**
NL-KP-09	7.04190	8.06275	**5.04180**
Preveza2008	0.00002	0.00002	0.00002
SecondarySchool2	0.00002	0.00002	0.00002
UK-SP-06	13.00636	13.00880	**13.00636**
UniversityInstance3	0.00006	0.00006	0.00006
UniversityInstance5	0.00000	0.00000	0.00000

US-WS-09	0.00518	0.00184	**0.00145**
VillageSchool	0.00013	0.00013	0.00013
ZA-WD-09	2.00000	2.00000	2.00000
ZL-LW-09	2.00034	**1.00034**	**1.00034**
Wins	-	**2**	**10**
Losses	-	**10**	**2**
Ties	-	**27**	**27**

Some of the datasets are large datasets (NL-KP-03, NL-KP-05, NL-KP-09, Kottenpark2008, GEPRO, DK-FG-12, DK-HG-12, DK-VG-09, KS-PR-11, and UK-SP-06), medium datasets (AU-BG-98, AU-TE-99, ES-SS-08, FI-WP-06, GR-PA-08, Aigio2010, IT-I4-96, US-WS-09, and ZA-WD-09) and for small datasets (ZL-LW-09, BrazilInstance5 and BR-SM-00). By comparing BGDA with MGDA, MGDA outperformed BGDA in 10 instances (DK-FG-12, FI-PB-98, GEPRO, Kottenpark2008, KS-PR-11, NL-KP-03, NL-KP-05, NL-KP-09, UK-SP-06 and US-WS-09), with the remaining 27 datasets were tied. However, distinct situation occurred to DK-VG-09 and DK-HG-12 which produced better result using BGDA compared to MGDA.

Based on the aforementioned findings, employing both hard and soft water level inputs in MGDA facilitated a smoother approach towards the acceptance threshold, leading to improved convergence towards better solution. The better results on two datasets that produced by BGDA compared to MGDA were perhaps due to randomness of selecting the NS.

Table 6.4 compares MGDA with several metaheuristic approaches which are VNS (Fonseca et al., 2016c), SA+ILS (Teixeira et al., 2019), SGVNS (Teixeira et al., 2019), GVNS (Teixeira et al., 2019), HSA-SA (Sørensen & Dahms, 2014) and SA-ILS (Fonseca et al., 2016a). However, out of 39 datasets, not all datasets for each methods provide results. In comparison to the metaheuristics, MGDA offers the best solutions

on 13 datasets (ArtificialSchool, AU-SA-96, AU-TE-99, BrazilInstance3, BrazilInstance5, BrazilInstance7, BR-SA-00, BR-SN-00, IT-I4-96, Kottenpark2008, NL-KP-09, Preveza2008 and VillageSchool), and ties on 5 datasets (ElementarySchool, GR-H1-97, GR-P3-10, KS-PR-11 and UniversityInstance5) out of 39 datasets. Using the average rank for assessing performance, MGDA posted a result of 2.08. When compared to other approaches based on average rank, MGDA comes in the second place, after the VNS and followed by SGVNS, GVNS, SA-ILS, HSA-SA and finally SA+ILS.

Table 6.5 compares MGDA with several state-of-the-art methods, including A) Mixed-Integer Linear Programming (Kristiansen et al., 2015b), B) Matheuristics (Fonseca et al., 2016c), C) MaxSAT-based large neighbourhood search (Demirović & Musliu, 2017), and D) Hidden Markov Model Approach to Heuristic Selection (Kheiri & Keedwell, 2017). In general, MGDA is successful in minimizing soft constraint violation and reasonable at minimizing hard constraint violations. In comparison to the state-of-the-art methods, the MGDA obtains the best results for 7 datasets (Aigio2010, BrazilInstance1, Kottenpark2008, Preveza2008, SecondarySchool2, UK-SP-06 and VillageSchool) and 4 ties (ElementarySchool, GR-H1-97, GR-P3-10 and UniversityInstance5). Of the ties, 3 of them has reached optimality. The average rank is also used to compare the performance of the four algorithms (A, B, C, and D) with MGDA. MGDA achieved a score of 2.31, placing it in the fourth position.

Table 6.6 demonstrates the result of hybrid DHSA with MGDA (DHSA + MGDA) that compared to DHSA results (Section 5.4). Out of 39 instances, hybridization of DHSA with MGDA outperformed DHSA in 9 instances (AU-BG-98, AU-TE-99, DK-FG-12,

DK-HG-12, NL-KP-05, NL-KP-09, Kottenpark2008, GEPRO and UK-SP-06), while produces worst results in 4 data instances (BrazilInstance5, DK-VG-09, IT-I4-96 and NL-KP-03), with the remaining of 26 datasets were tied.

Table 6.4

Results Obtained Using MGDA Compared to Metaheuristics-Based Approaches

Instance	MGDA	VNS	SA + ILS	SGVNS	GVNS	HAS-SA	SA-ILS
Aigio2010	0.00006	-	-	-	-	-	**0.00000**
ArtificialSchool	**3.00006**	-	-	-	-	-	-
AU-BG-98	1.00469	2.00398	6.00450	**1.00401**	4.00370	-	-
AU-SA-96	**0.00005**	4.00021	14.00050	13.00046	12.00051	-	-
AU-TE-99	**0.00085**	1.00036	7.00161	7.00163	7.00151	-	-
BrazilInstance1	0.00048	-	0.00012	**0.00011**	**0.00011**	0.00020	-
BrazilInstance3	**0.00065**	-	-	-	-	0.00154	0.00101
BrazilInstance5	**0.00068**	-	0.00164	0.00149	0.00158	0.00148	-
BrazilInstance7	**0.00130**	-	0.00264	0.00248	0.00249	0.00234	-
BR-SA-00	**0.00011**	0.00022	-	-	-	0.00048	0.00032
BR-SM-00	2.00119	3.00099	0.00091	**0.00090**	0.00094	0.00162	1.00136
BR-SN-00	**0.00096**	0.00104	0.00149	0.00131	0.00148	0.00186	0.00160
DK-FG-12	0.01907	**0.01669**	-	-	-	-	-
DK-HG-12	12.03664	**12.03371**	-	-	-	-	-
DK-VG-09	2.03584	**2.02765**	-	-	-	-	-
ElementarySchool	**0.00003**	-	-	-	-	**0.00003**	**0.00003**
ES-SS-08	0.00555	0.00415	-	-	-	-	**0.00012**
FI-MP-06	0.00091	**0.00084**	0.00086	0.00088	0.00087	0.00090	-
FI-PB-98	0.00005	**0.00000**	-	-	-	-	-
FI-WP-06	0.00013	0.00011	**0.00001**	**0.00001**	**0.00001**	-	-
GEPRO	1.00613	-	1.00566	1.00441	**1.00434**	-	-
GR-H1-97	**0.00000**	**0.00000**	-	-	-	-	-
GR-P3-10	**0.00000**	**0.00000**	-	-	-	-	-
GR-PA-08	0.00005	**0.00003**	-	-	-	-	0.00008
IT-I4-96	**0.00040**	0.00042	-	-	-	0.00082	0.00061
Kottenpark2008	**14.34855**	-	-	-	-	-	-
KS-PR-11	**0.00004**	**0.00004**	-	-	-	-	-
NL-KP-03	0.01376	**0.01151**	0.01409	0.01281	0.01216	-	0.05355
NL-KP-05	1.01850	8.04460	0.01078	**0.00877**	0.00881	-	24.13930
NL-KP-09	**5.04180**	8.08370	-	-	-	-	19.05565
Preveza2008	**0.00002**	-	-	-	-	-	-
SecondarySchool2	0.00002	-	-	-	-	**0.00000**	**0.00000**
UK-SP-06	13.00636	53.01524	0.18092	**0.12466**	0.12542	-	-
UniversityInstance3	0.00006	-	-	-	-	-	**0.00005**

UniversityInstance5	**0.00000**	-	-	-	-	-	**0.00000**
US-WS-09	0.00145	**0.00124**	-	-	-	-	-
VillageSchool	**0.00013**	-	-	-	-	-	-
ZA-WD-09	2.00000	3.00000	-	-	-	-	**0 .00441**
ZL-LW-09	1.00034	**0.00016**	0.00022	0.00024	0.00024	-	-
Rank Average	**2.08**	**2.04**	**3.40**	**2.40**	**2.67**	**3.27**	**2.75**
Rank	**2**	**1**	**7**	**3**	**4**	**6**	**5**

Note: *Dash (-)* denotes that there are no results provided by the author.

Table 6.5

Results Obtained Using MGDA Compared to The-State-Of-The-Art Methods

Instance	**MGDA**	**A**	**B**	**C**	**D**
Aigio2010	**0.00006**	-	-	0.04582	-
ArtificialSchool	3.00006	-	-	**0.00012**	-
AU-BG-98	1.00469	3.00494	2.00398	-	**0.00520**
AU-SA-96	0.00005	8.00052	3.00021	-	**0.00002**
AU-TE-99	0.00085	1.00140	1.00036	-	**0.00061**
BrazilInstance1	**0.00048**	-	-	-	-
BrazilInstance3	0.00065	**0.00026**	-	0.00075	-
BrazilInstance5	0.00068	**0.00030**	-	0.00224	-
BrazilInstance7	0.00130	**0.00122**	-	0.00603	-
BR-SA-00	0.00011	**0.00005**	0.00005	0.00057	0.00010
BR-SM-00	2.00119	0.00061	**0.00052**	0.00214	2.00117
BR-SN-00	0.00096	0.00060	**0.00035**	0.00352	0.00101
DK-FG-12	0.01907	2.23705	0.01668	-	**0.01522**
DK-HG-12	12.03664	293.3211	12.03371	-	**12.0263**
DK-VG-09	2.03584	20.18966	2.02765	-	**2.02731**
ElementarySchool	**0.00003**	-	-	0.00003	-
ES-SS-08	0.00555	0.00357	**0.00351**	-	0.00517
FI-MP-06	0.00091	0.00088	**0.00077**	0.00504	0.00089
FI-PB-98	0.00005	-	**0.00000**	0.01309	0.00008
FI-WP-06	0.00013	**0.00001**	0.00002	0.00812	0.00007
GEPRO	1.00613	**1.00566**	-	-	-
GR-H1-97	**0.00000**	-	**0.00000**	**0.00000**	**0.00000**
GR-P3-10	**0.00000**	-	**0.00000**	0.02329	**0.00000**
GR-PA-08	0.00005	0.00008	**0.00003**	0.00141	0.00004
IT-I4-96	0.00040	**0.00027**	**0.00027**	0.16979	0.00038
Kottenpark2008	**14.34855**	-	-	-	-
KS-PR-11	0.00004	0.00003	**0.00000**	0.29946	0.00003
NL-KP-03	0.01376	0.01410	0.01103	-	**0.00466**
NL-KP-05	1.01850	0.01078	8.04460	-	**0.00811**
NL-KP-09	5.04180	**0.09035**	7.64470	-	2.07495
Preveza2008	**0.00002**	-	-	0.05617	-
SecondarySchool2	**0.00002**	-	-	0.03523	-

UK-SP-06	**13.00636**	-	53.01524	-	19.01294
UniversityInstance3	0.00006	**0.00005**	-	0.00007	-
UniversityInstance5	**0.00000**	-	-	**0.00000**	-
US-WS-09	0.00145	-	**0.00124**	-	0.00512
VillageSchool	**0.00013**	-	-	-	-
ZA-WD-09	2.00000	**0.00000**	**0.00000**	**0.00000**	9.00000
ZL-LW-09	1.00034	-	**0.00000**	0.01039	0.00052
Rank Average	**2.31**	**2.25**	**1.86**	**2.95**	**2.04**
Rank	**4**	**2**	**1**	**5**	**3**

Note: *Dash (-)* denotes that there are no results provided by the author

Table 6.7 indicates comparison of DHSA+MGDA and DHSA with several state-of-the-art approaches, including A) Mixed-Integer Linear Programming (Kristiansen et al., 2015b), B) Matheuristics (Fonseca et al., 2016c), C) MaxSAT-based large neighborhood search (Demirović & Musliu, 2017), and D) Hidden Markov Model Approach to Heuristic Selection (Kheiri & Keedwell, 2017). DHSA+MGDA secured a score of 2.20, earning third place, whereas DHSA recorded 2.41, landing it in the fourth position. DHSA+MGDA produced the best results on 4 datasets (AU-TE-99, Kottenpark2008, GEPRO, and UK-SP-06) and tied with 4 instances (ElementarySchool, GR-H1-97, GR-P3-10 and UniversityInstance5). In addition, the community[8] of the High School Timetabling, DHSA+MGDA produced the best results (1.00547) so far for large dataset named as GEPRO from Netherlands.

Regarding adaptive parameters, DHSA+MGDA is similar in DHSA that has a chance of pitch adjustment rate due to MGDA's excellent ability to diversify over two decay rates. At this moment, the DHSA's adaptive parameters had totally and adaptively blended with the MGDA's threshold water. Due to this similarity of adaptive parameters tuning of DHSA+MGDA with DHSA, this Chapter will not further discuss the

[8] High School Timetabling Project (HSTT)
https://www.utwente.nl/en/eemcs/dmmp/hstt/datasets/Netherlands/GEPRO/

DHSA+MGDA adaptive parameters tuning analysis as Section 5.5 already discussed it in detail. Instead, the following section delves into the impact analysis of water level parameters on BGDA and MGDA.

Table 6.6

The Hybridization DHSA with MGDA and DHSA Results

Instance	DHSA + MGDA	DHSA
AU-BG-98	**1.00439**	1.00446
AU-SA-96	0.00003	0.00003
AU-TE-99	**0.00041**	0.00075
BR-SA-00	0.00011	0.00011
BR-SM-00	2.00119	2.00119
BR-SN-00	0.00096	0.00096
BrazilInstance5	0.00071	**0.00068**
BrazilInstance3	0.00065	0.00065
BrazilInstance7	0.00134	0.00134
BrazilInstance1	0.00048	0.00048
VillageSchool	0.00013	0.00013
DK-FG-12	**0.01730**	0.01832
DK-HG-12	**12.03323**	12.03419
DK-VG-09	2.03348	**2.03327**
ES-SS-08	0.00542	0.00542
FI-PB-98	0.00005	0.00005
FI-WP-06	0.00013	0.00013
FI-MP-06	0.00091	0.00091
SecondarySchool2	0.00000	0.00000
ElementarySchool	0.00003	0.00003
ArtificialSchool	3.00004	3.00004
GR-H1-97	0.00000	0.00000
GR-P3-10	0.00000	0.00000
GR-PA-08	0.00005	0.00005
UniversityInstance3	0.00006	0.00006
UniversityInstance5	0.00000	0.00000
Preveza2008	0.00002	0.00002
Aigio2010	0.00004	0.00004
IT-I4-96	0.00039	**0.00038**
KS-PR-11	0.00004	0.00004
NL-KP-03	0.00859	**0.00806**
NL-KP-05	**3.01142**	3.01143
NL-KP-09	**5.04070**	5.04190
Kottenpark2008	**13.36491**	14.32969
GEPRO	**1.00547**	1.00567
UK-SP-06	**7.00810**	11.00904
US-WS-09	0.00126	0.00126

ZL-LW-09	0.00004	0.00004
ZA-WD-09	2.00000	2.00000
Wins	**9**	**4**
Losses	**4**	**9**
Ties	**26**	**26**

Table 6.7

Results of Hybridization DHSA with MGDA and DHSA Compared to The State-Of-The-Art Methods

Instance	DHSA+MGDA	DHSA	A	B	C	D
AU-BG-98	1.00439	1.00446	3.00494	2.00398	-	**0.00520**
AU-SA-96	0.00003	0.00003	8.00052	3.00021	-	**0.00002**
AU-TE-99	**0.00041**	0.00075	1.00140	1.00036	-	0.00061
BR-SA-00	0.00011	0.00011	**0.00005**	0.00005	0.00057	0.00010
BR-SM-00	2.00119	2.00119	0.00061	**0.00052**	0.00214	2.00117
BR-SN-00	0.00096	0.00096	0.00060	**0.00035**	0.00352	0.00101
BrazilInstance5	0.00071	0.00068	**0.00030**	-	0.00224	-
BrazilInstance3	0.00065	0.00065	**0.00026**	-	0.00075	-
BrazilInstance7	0.00134	0.00134	**0.00122**	-	0.00603	-
BrazilInstance1	0.00048	0.00048	-	-	-	-
VillageSchool	0.00013	**0.00013**	-	-	-	-
DK-FG-12	0.01730	0.01832	2.23705	0.01668	-	**0.01522**
DK-HG-12	12.03323	12.03419	293.3211	12.03371	-	**12.02628**
DK-VG-09	2.03348	2.03327	20.18966	2.02765	-	**2.02731**
ES-SS-08	0.00542	0.00542	0.00357	**0.00351**	-	0.00517
FI-PB-98	0.00005	0.00005	-	**0.00000**	0.01309	0.00008
FI-WP-06	0.00013	0.00013	**0.00001**	0.00002	0.00812	0.00007
FI-MP-06	0.00091	0.00091	0.00088	**0.00077**	0.00504	0.00089
SecondarySchool2	0.00000	**0.00000**	-	-	0.03523	-
ElementarySchool	0.00003	0.00003	-	-	0.00003	-
ArtificialSchool	3.00004	3.00004	-	-	**0.00012**	-
GR-H1-97	0.00000	**0.00000**	-	**0.00000**	**0.00000**	**0.00000**
GR-P3-10	0.00000	**0.00000**	-	**0.00000**	0.02329	**0.00000**
GR-PA-08	0.00005	0.00005	0.00008	**0.00003**	0.00141	0.00004

Instance	DHSA+MGDA	DHSA	A	B	C	D
UniversityInstance3	0.00006	0.00006	**0.00005**	-	0.00007	-
UniversityInstance5	0.00000	**0.00000**	-	-	**0.00000**	-
Preveza2008	0.00002	**0.00002**	-	-	0.05617	-
Aigio2010	0.00004	**0.00004**	-	-	0.04582	-
IT-I4-96	0.00039	0.00038	**0.00027**	**0.00027**	0.16979	0.00038
KS-PR-11	0.00004	0.00004	0.00003	**0.00000**	0.29946	0.00003
NL-KP-03	0.00859	0.00806	0.01410	0.01103	-	**0.00466**
NL-KP-05	3.01142	3.01143	0.01078	8.04460	-	**0.00811**
NL-KP-09	5.04070	5.04190	**0.09035**	7.64470	-	2.07495
Kottenpark2008	**13.36491**	14.32969	-	-	-	-
GEPRO	**1.00547**	1.00567	1.00566	-	-	-
UK-SP-06	**7.00810**	11.00904	-	53.01524	-	19.01294
US-WS-09	0.00126	0.00126	-	**0.00124**	-	0.00512
ZL-LW-09	0.00004	0.00004	-	**0.00000**	0.01039	0.00052
ZA-WD-09	2.00000	2.00000	**0.00000**	**0.00000**	**0.00000**	9.00000
Rank Average	**2.20**	**2.41**	**2.50**	**2.04**	**3.04**	**2.16**
Rank	**3**	**4**	**5**	**1**	**6**	**2**

Note: *Dash (-)* denotes that there are no results provided by the author.

6.6 Analysis and Discussion on Water Levels of BGDA and MGDA

Figures 6.6, 6.7, 6.8, 6.9, 6.10 and 6.11 highlight the old solution of timetable (blue line colour), new solution of timetable (orange line colour) and water level parameter (grey line colour). Figure 6.6 shows US-WS-09 dataset that start in iteration 1 until 5 where the abnormal drop of water level parameter due to the fix value of parameter settings (*rainspeed* and *totaIter*) in the beginning of search space. Beginning with iteration 6, MGDA adopts an adaptive approach where the soft water level parameter is dynamically computed. A new solution is then determined based on this parameter and the old soft solution. This process is similar with Figure 6.7, where, for each iteration using the NL-KP-03 dataset, the new solution is picked in association with the soft water level parameter and the old soft solution.

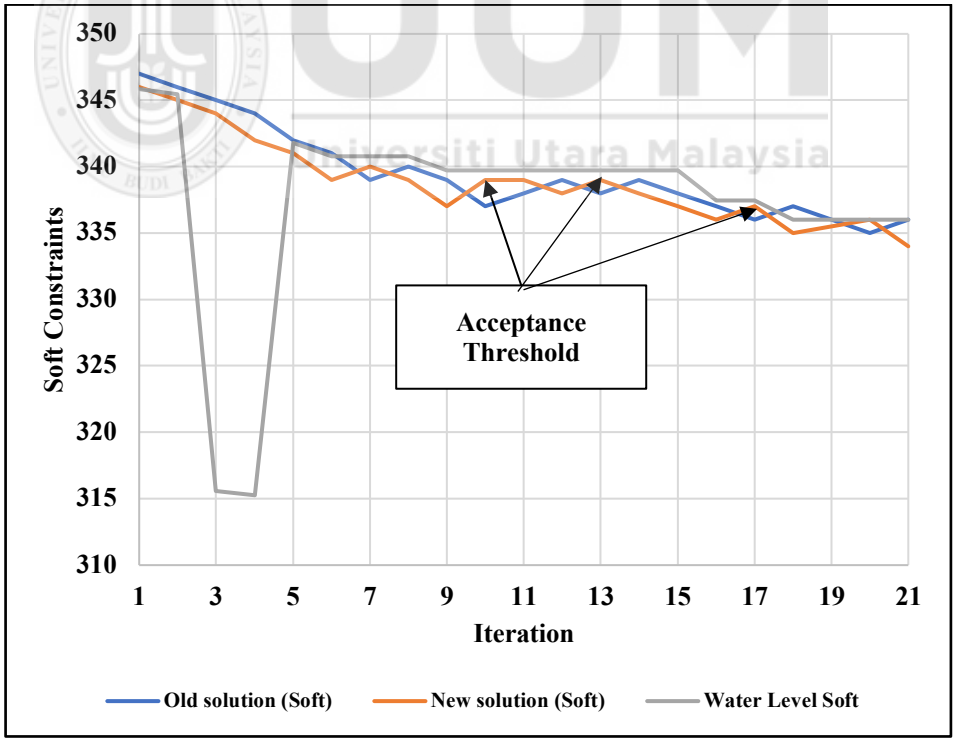


Figure 6.6. The soft water level for *US-WS-09* using MGDA

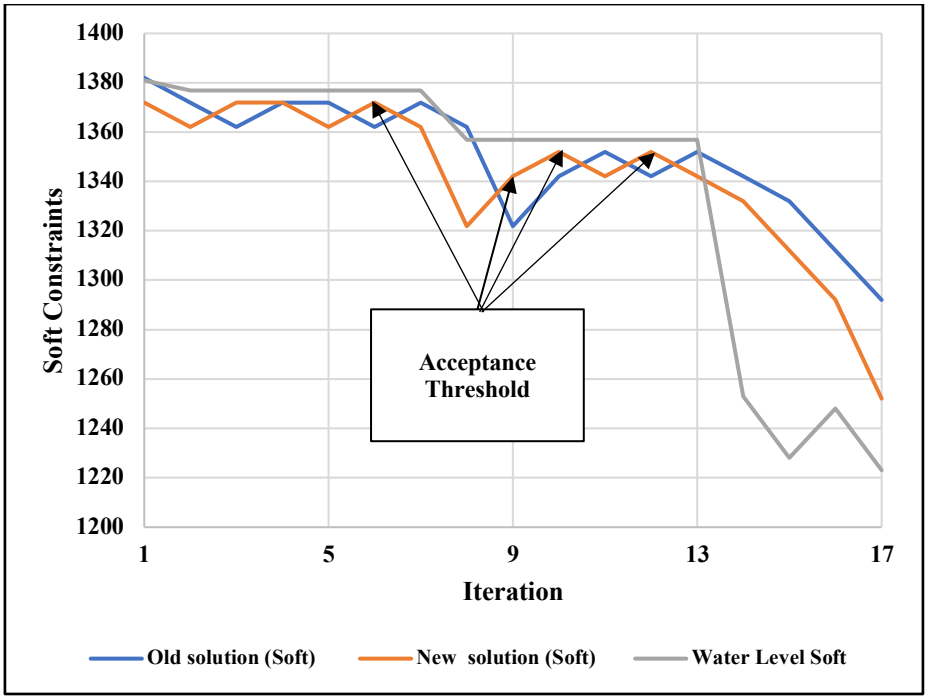


Figure 6.7. The soft water level for *NL-KP-03* using MGDA

The differences between the two techniques (MGDA and BGDA) in avoiding local optima can be seen in Figures 6.8 and 6.9, respectively. In Figure 6.9, it is obviously shown that the new soft solution of MGDA was in corresponding with the soft water level parameter and old soft solution starting from first iteration. However, in Figure 6.9, the BGDA is inconsistent in terms of water level in which it drops tremendously starting from iteration 13 to 17. In addition, the BGDA unable to reach the acceptance threshold (the new solution is out of alignment with the water level parameter and the old solution).

Figure 6.8. The soft water level for *AU-BG-98* using MGDA

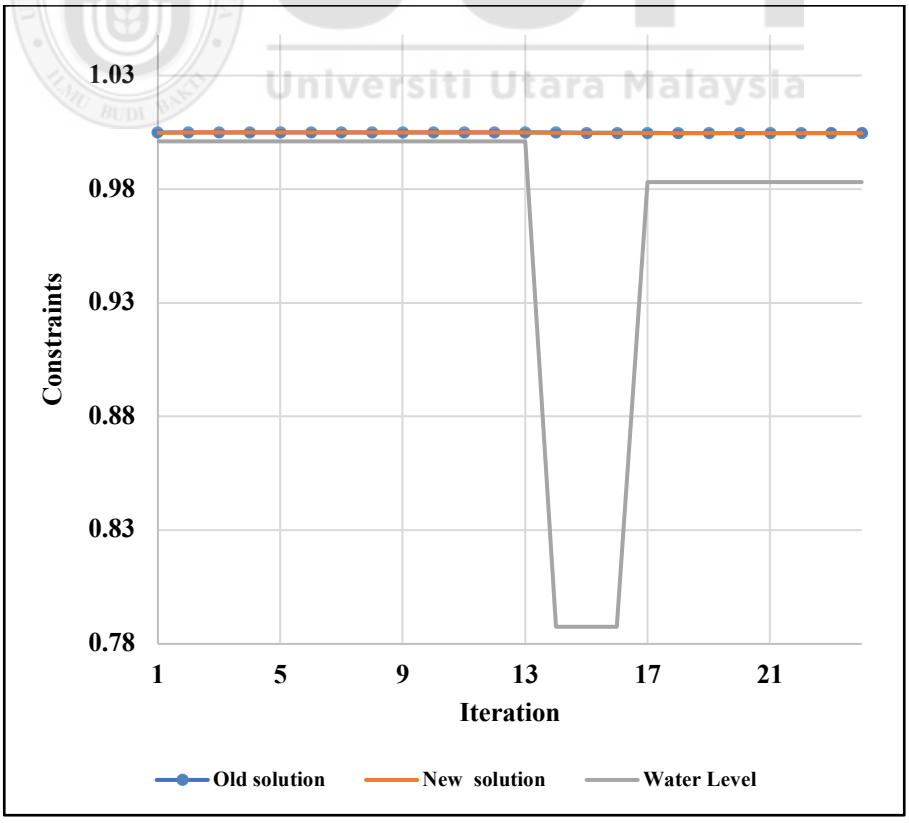


Figure 6.9. The water level for *AU-BG-98* using BGDA

Figure 6.10 shows the MGDA with soft and hard water levels using Kottenpark2008 data instance. The Figure 6.10 (a) shows similar pattern as in Figures 6.6, 6.7 and 6.8 which the new soft solution is consistent with soft water level and old soft solution. In addition, the new soft or hard solution was accepted corresponding on the acceptance of new hard or soft solution from iterations 1 to 16 (Figure 6.10 (a) and (b), respectively) in order to match with the old solution (soft or hard) and water levels (soft or hard). For example, in iteration 2, the worst new hard solution was accepted because there was a good new soft solution. However, in Figure 6.11, the BGDA shows inconsistency with the water level from iteration 9 onwards, failing to meet the acceptance threshold. The new solution deviates from both the water level parameter and the previous solution.

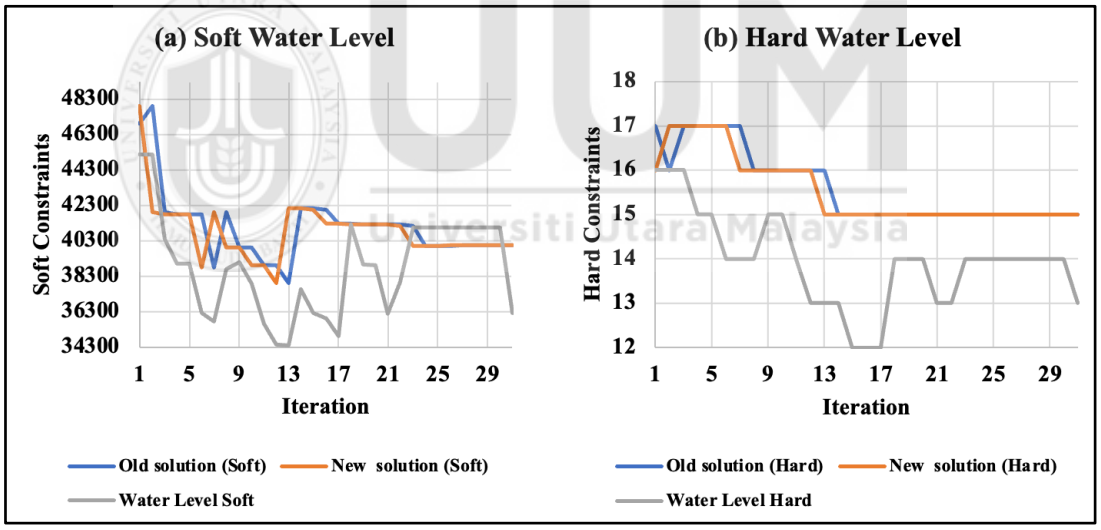


Figure 6.10. The (a) soft and (b) Hard water level for *Kottenpark2008* using MGDA

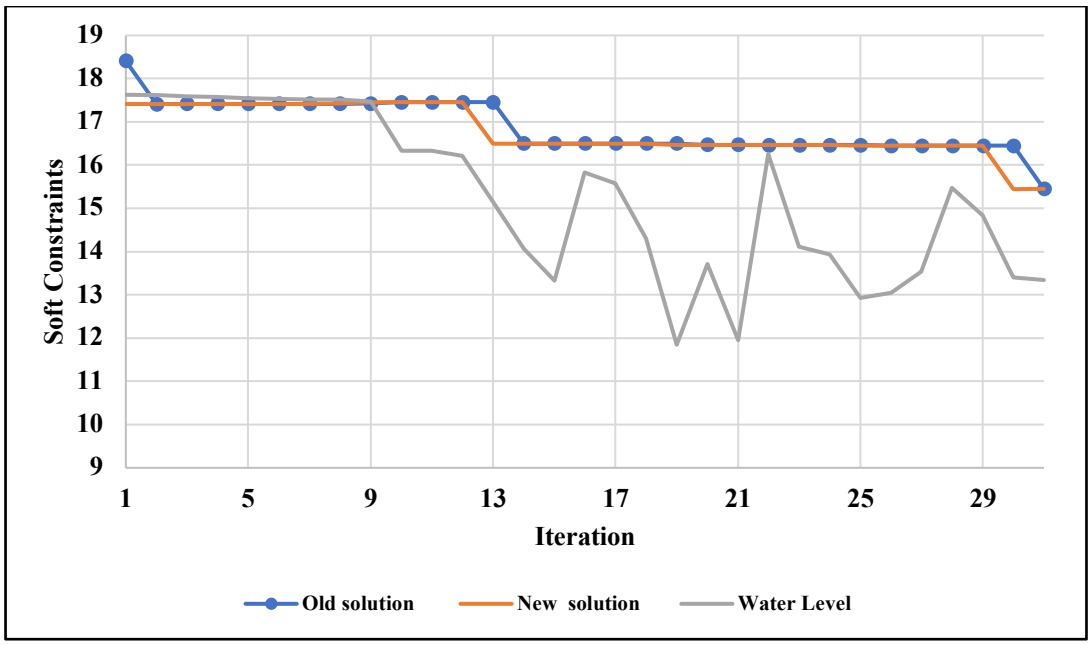


Figure 6.11. The water level for *Kottenpark2008* using BGDA

6.6.1 Analysis of Water Level Measurement for MGDA and Previous Adaptive GDA

This section provides a comparison and analysis of the objective function produced by MGDA and the previous adaptive GDA methods (Burke et al., 2006; Landa-Silva & Obit, 2008a; Mushi, 2011) which used the AU-TE-99 dataset to execute the experiment with different iteration sizes (50, 100 and 200), as shown in Table 6.8. The aim is to highlight the use of water levels with the objective function for both methods: MGDA and the previous adaptive GDA, focusing on dual and non-linearity decay rates, respectively. In producing objective function of AU-TE-99 dataset, decay rate in Equation 2.19 was used to calculate water level in each iteration. For setting the parameters of δ, β, min and max, the value of δ is assigned to medium instance as AU-TE-99 dataset is medium size. The value of β was set to 0, as set in Chapter Two. The values of min and max were chosen as 100,000 and 300,000, respectively. In Table 6.8, the previous methods resulted in a high-cost objective function of 29211.00480,

comprising 29211 hard constraints and 480 soft constraints. This high cost is due to the isolation of water levels from the objective function, where the water level rate is calculated solely based on fixed parameters related to dataset size. On the other hand, MGDA reduces the number of soft constraints by 2, achieving 85 soft constraints from the initial solution for each iteration size.

Table 6.8

Comparison of objective funcation obtained by MGDA and previous adaptive GDA using AU-TE-99 dataset across different iteration sizes.

Iteration	Initial Solution	MGDA	Previous Adaptive GDA
50		0.00085	29211.00480
100	0.00087	0.00085	29211.00480
200		0.00085	29211.00480

6.6.2 Discussion on Hybridization of DHSA with MGDA

This chapter proposed two enhancements to the methodology for solving HSTP. The first enhancement is on addressing the limitations of the basic GDA by improving the handling of both soft and hard constraints separately since the basic GDA lacks the ability to independently and effectively address soft and hard constraints, often resulting in suboptimal or totally wores (with hard constraits) solutions. The MGDA was introduced to address this gap by implementing separate water levels and decay rates for each constraint type, thereby enabling a more balanced and targeted optimization approach. This enhancement solves the issue of insufficient control over the water level adjustment process, as MGDA introduces separate water levels and decay rates for each constraint type, allowing for a more balanced and targeted approach to optimization. The second enhancement is to introduce a hybrid approach

by hybridizing DHSA with MGDA, to improve the hybrid approach by combining the exploration capabilities of DHSA with the exploitation strengths of MGDA. This integration enhances the algorithm's ability to escape local optima and provides better adaptability in handling diverse problem instances. The experimental results presented in Section 6.5 demonstrate that both enhancements (i.e. the MGDA and the hybridized method) consistently outperforms the basic GDA, particularly on large and complex datasets, where the original algorithm struggled. MGDA improved the methodology by introducing two adaptive water levels and decay rates, which better manage soft and hard constraints independently, resulting in more effective constraint satisfaction. The hybrid method improved by leveraging MGDA's adaptive features and integrating them into DHSA's adaptive parameters, thereby achieving a balance between diversification and intensification for superior optimization performance. The hybridization enhances diversification in the search process, allowing the algorithm to avoid local optima and explore more of the solution space. Statistical comparisons against other metaheuristic approaches show that DHSA combined with MGDA produces higher-quality solutions, confirming that the enhancement not only solves the issue but also provides superior performance across a variety of benchmark datasets.

To measure whether local optima and diversification issues have been addressed, it is essential to evaluate specific indicators and analyze the performance of the algorithm comprehensively. MGDA employs mechanisms such as adaptive water level adjustments to escape local optima and improve the diversification of the search process. These claims can be substantiated through a detailed examination of convergence metrics, solution quality improvements, and diversity indices. First, the ability of MGDA to avoid local optima can be evaluated by analyzing its convergence

behavior. Metrics such as the variance in solution quality across iterations and the frequency of improvements demonstrate whether the algorithm can overcome stagnation. For example, tracking the performance on benchmark datasets can reveal instances where MGDA achieves superior solutions compared to standard algorithms. If the MGDA consistently produces improvements even when other algorithms plateau, this indicates its capability to escape local optima effectively. Second, diversification is measured by evaluating the variety of solutions explored during the search process. Hybridization changes in solution characteristics over iterations provides insights into how well the algorithm explores the solution space. Additionally, monitoring transitions between different neighborhoods validates that the algorithm avoids premature convergence and maintains robust exploration. The adaptive parameters in MGDA, combined with a water level decay mechanism, encourage the algorithm to explore new areas of the solution space, ensuring that it does not get trapped in suboptimal regions. Additionally, analyzing transitions between diverse neighborhoods further validates that the algorithm avoids premature convergence and maintains robust exploration. Empirical evidence from benchmark datasets supports the claims of improved diversification and local optima avoidance. For instance, performance comparisons across algorithms like BGDA, MGDA, and traditional methods show how MGDA achieves better satisfaction of hard and soft constraints in high school timetabling problems. This is particularly evident in cases where MGDA successfully navigates complex solution spaces that pose challenges for other methods. The adaptive water level adjustment, which incorporates the problem's constraints, allows MGDA to relax acceptance criteria dynamically, enabling it to bypass local optima and enhance its exploration capabilities.

6.7 Summary

This chapter presents an implementation of basic and modified great deluge metaheuristic for solving HSTP. Computational tested on well-known benchmark datasets verified MGDA effectiveness. The MGDA's effectiveness was evaluated based on decay rates and compared to BGDA, other metaheuristics, and state-of-the-art methods using HSTP benchmark datasets. MGDA with two decay rates is effective at producing timetables of good quality. It efficiently guides the search through each neighbourhood structure, leveraging two decay rates to avoid local optima and ensure diversification. MGDA was ranked as the second best among metaheuristics methods and the fourth best among state-of-the-art methods. Overall, the MGDA was able to provide satisfactory results especially on large datasets. When MGDA and DHSA were hybridized, the population solution and adaptive parameters of DHSA were fully integrated with MGDA threshold phenomena due to the DHSA pitch adjustment rate. DHSA with MGDA algorithms produced HTSP solutions of excellent quality when compared to state-of-the-art methods. The water level analysis of MGDA is compared to previous adaptive GDA methods which highlights the superior performance of MGDA against previous adaptive GDA methods. Finally, a discussion on the proposed enhancement and hybridization addresses the problem with GDA water level or threshold acceptance rules, which are not adaptive to the behavioral neighborhood moves of GDA and therefore need to be hybridized with PAR in DHSA. Chapter Seven will describe the procedure of generating a real-world school timetabling problem (STP) and solving it using construction (Chapter Four) and improvement methods (Chapters Five and Six).

CHAPTER SEVEN

PRODUCING AND SOLVING REAL-WORLD DATASET OF SCHOOL TIMETABLING PROBLEM

7.1 Introduction

This chapter presents the procedure of producing and solving a real-world dataset of Malaysian school timetabling problem (STP). The real-world dataset was obtained from STP timetable of a primary school located in the north of Kedah, Malaysia. It was converted to STP dataset based on the XHSTT dataset format that has the same constraints as depicted in XHSTT dataset. The real-world dataset was categorized into three difficulty levels which are easy, normal and hard based on the difficulty level of preassigned teachers. Next, the real-world dataset was solved using the MCHR construction method to produce the initial solution and then the solution by MCHR was improved with MGDA and DHSA methods. Firstly, Section 7.2 illustrates the step by step of converting the STP timetable to XHSTT data instance. Next, Section 7.3 highlights the experimental and results of solving the real-world XHSTT data instance. Finally, summary of the chapter is presented in Section 7.4.

7.2 Converting STP Timetable to XHSTT Instances

The conversion process is composed of extracting information from the STP timetable related to components in the XHSTT format i.e., metadata, times, resources, events, and constraints. The STP timetable in this study was obtained from the primary school located at Bukit Kayu Hitam, Kedah, Malaysia. The STP timetable is generated for the year 2021.

7.2.1 Metadata

The first component of the XHSTT instance is the metadata which is the information about data. The information consists of name, contributor, date, country, and description. The metadata for the STP timetable is shown in Figure 7.1.

```
<MetaData>
    <Name>SFBT21</Name>
    <Contributor>Universiti Utara Malaysia, College of Arts and Sciences, School of Computing</Contributor>
    <Date>26 August 2021</Date>
    <Country>Malaysia</Country>
    <Description>SK FELDA BUKIT TANGGA JADUAL WAKTU KELAS 2021</Description>
</MetaData>
```

Figure 7.1. Metadata View

7.2.2 Time

The second component is time. Times consists of two information which are *Timegroups* (sets of times), and *Time Id* (the times of the instance). The *Timegroups* consist of the days of weeks, while the *Time Id* represents one slot (one duration) of day. As shown in Figure 7.2, the days of the STP timetable consist of Sunday, Monday, Tuesday, Wednesday, and Thursday. These days were converted to the *Timegroups* as *Day_1, Day_2, Day_3, Day_4,* and *Day_5* respectively (as shown by the red color boxes and arrow). The STP timetable also consists of 12 timeslots. These timeslots were converted to *Time Id* that contains the *Name* and *Day Reference*. For example, as shown with blue/green boxes and arrows in Figure 7.2, timeslot 1 on Sunday is converted to *Su1* (*Name*) and *Day_1* (*Day Reference*), while the timeslot 3 on Sunday is converted to *Su3* (*Name*) and *Day_1* (*Day Reference*). All 12 timeslots for each day (5 days) were converted to *Name* and *Day Reference* accordingly. A total of 60 *Time Id* was produced for the STP timetable.

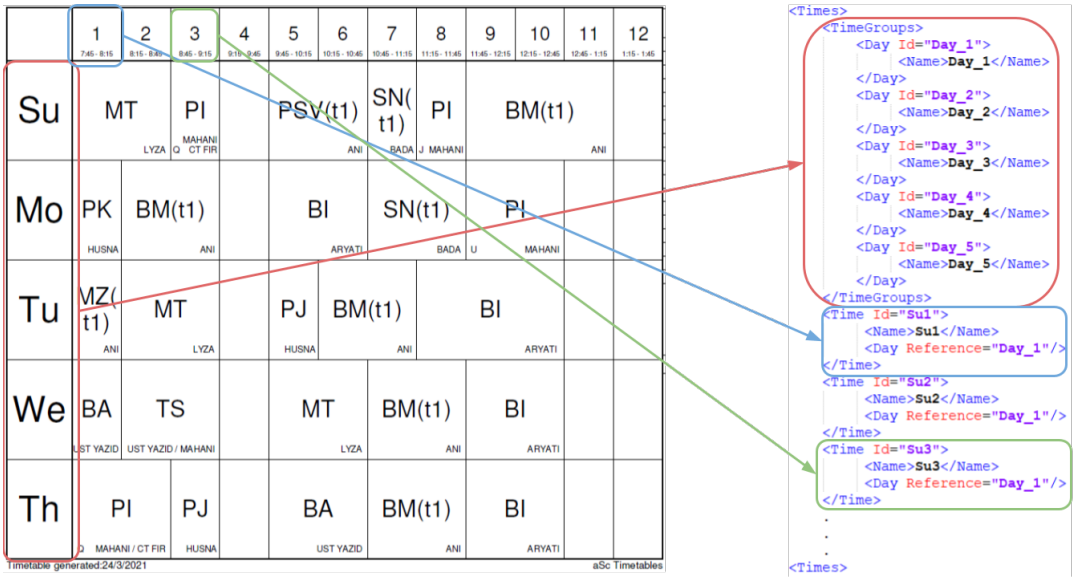


Figure 7.2. Conversion of STP Timetable to *Times* Component

7.2.3 Resources

The third component is resources. First of all, the *ResourceTypes* for the STP timetable were encoded as shown in Figure 7.3. Based on the STP timetable, the *ResourceTypes* consists of teacher and class. There were 25 teachers and 13 Classes in the STP timetable.


```
<Resources>
    <ResourceTypes>
        <ResourceType Id="Teacher">
            <Name>Teacher</Name>
        </ResourceType>
        <ResourceType Id="Class">
            <Name>Class</Name>
        </ResourceType>
    </ResourceTypes>
    .
    .
    .
</Resources>
```

Figure 7.3. Resource Type View

Next, all the teachers and classes in the STP timetable were converted to *ResourceGroups* as shown in Figure 7.4. The *ResourceGroups* were represented by

ResourceGroup Id, *Name,* and *ResourceType Reference*. The first *ResourceGroup Id* for the teacher is *Teacher_All* and other *ResourceGroup Id* were named based on the subject assigned to the teacher such as *Teacher_BATeacher_Div* (teacher that thought the Arabic Language), *Teacher_BM6ITeacher_Div* (teacher that thought the Malay Language), and so on. The first *ResourceGroup Id* for the class is *Class_All* and other *ResourceGroup Id* were named based on the class name such as *Class_ILTIZAM, Class_DINAMIK*, *Class_PRASEKOLAH,* and so on.

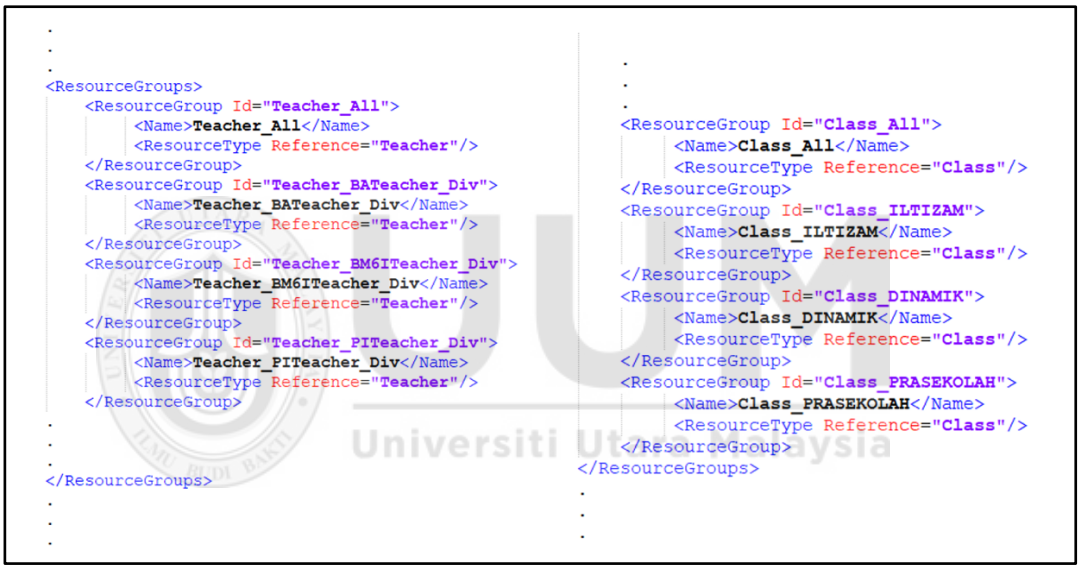


Figure 7.4. Resource Groups of Teacher and Class View

7.2.4 Events

The events are the combination of times and resources that consist of event groups (sets of events) (Figure 7.5) and the event itself (Figure 7.6). The event groups were defined by the *Course* and *EventGroup* as shown in Figure 7.5. *The course* consists of a combination between *Id* and *Name*. Based on the STP timetable, a total of 145 courses were listed. The same number of *EventGroup* was listed based on the *Course* as shown in Figure 7.5. The *Event* as shown in Figure 7.6 consists of *Id*, *Name*, *Duration*, *Course*

Reference, *Resources,* and *EventGroup Reference.* This is the part that combines the course (Classes and subjects) with the resources such as teacher and class. Based on the STP timetable, a total of 322 events and 578 durations were identified. For example, an event is represented in the red box as shown in Figure 7.6 contains the following such as the *Id* is *"1ILTIZAM_MT_1"*, the *Name* is *"1ILTIZAM_MT_1"*, *Duration* is *6, Course Reference* is *"1ILTIZAM_MT"*, *Resources* are *"MTTeacher03"* and *"1ILTIZAM"*, and *EventGroup Reference* are *"ALLEvents"* and *"LinkedTo_1ILTIZAM MT_1"*.

```
<Events>
    <EventGroups>
        <Course Id="1ILTIZAM_MT">
            <Name>1ILTIZAM_MT</Name>
        </Course>
        <Course Id="1ILTIZAM_PI">
            <Name>1ILTIZAM_PI</Name>
        </Course>
        <Course Id="1ILTIZAM_PSVt1">
            <Name>1ILTIZAM_PSVt1</Name>
        </Course>
        <Course Id="1ILTIZAM_SNt1">
            <Name>1ILTIZAM_SNt1</Name>
        </Course>
        .
        .
        .
        .
    </EventGroups>
</Events>
```

```
.
.
.
<EventGroup Id="LinkedTo_1ILTIZAM_MT_1">
    <Name>LinkedTo_1ILTIZAM_MT_1</Name>
</EventGroup>
<EventGroup Id="LinkedTo_1ILTIZAM_PI_1">
    <Name>LinkedTo_1ILTIZAM_PI_1</Name>
</EventGroup>
<EventGroup Id="LinkedTo_1ILTIZAM_PSVt1_1">
    <Name>LinkedTo_1ILTIZAM_PSVt1_1</Name>
</EventGroup>
<EventGroup Id="LinkedTo_1ILTIZAM_SNt1_1">
    <Name>LinkedTo_1ILTIZAM_SNt1_1</Name>
</EventGroup>
.
.
.
```


Figure 7.5. Course and Event Group View

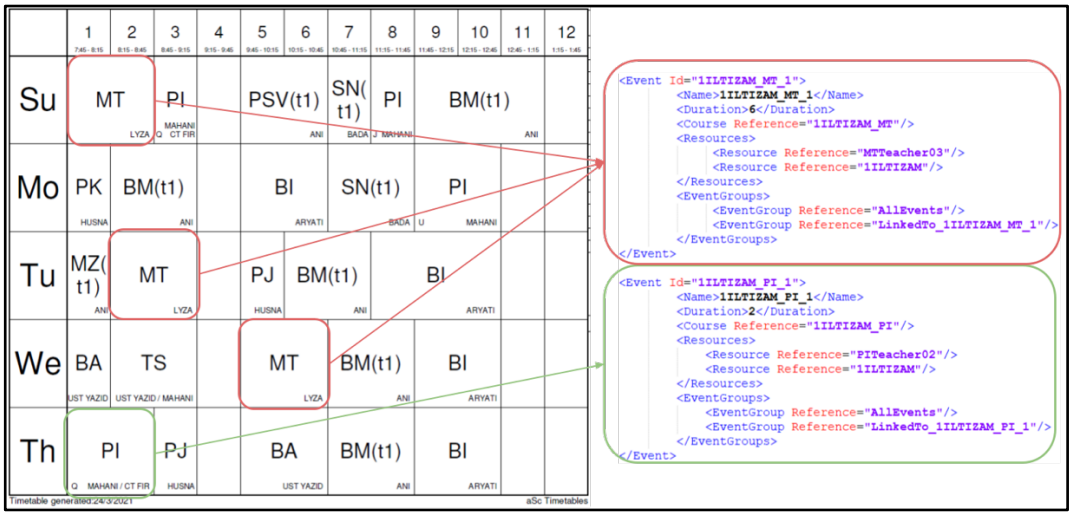


Figure 7.6. Conversion of STP Timetable to *Event* Component

7.2.5 Constraints

The constraint is a requirement that must be satisfied in order to solve the STP. The constraints of the XHSTT dataset should be presented as three categories which are scheduling, events, and resources constraints. Each schedule in the STP timetable was inspected in order to generate the three types of constraints. A total of 8 constraints with their total occurrences was identified from the STP timetable as shown in Table 7.1. All the constraints were also classified as hard or soft.

Table 7.1

The Constraints of SK FELDA BUKIT TANGGA

Type	Constraints	Hard/ Soft	Total
Scheduling	Assign Resource Constraint	Hard	1
	Assign Time Constraint	Hard	1
	Split Events Constraint	Hard	7
Events	Distribute Split Events Constraint	Hard	7
Resources	Avoid Clashes Constraint	Hard	1
constraints	Avoid Unavailable Times Constraint	Hard	5
	Limit Busy Times Constraint	Soft	89
	Limit Workload Constraint	Hard	17

As an example, the Limit Busy Times constraint from the STP timetable was produced based on Figure 7.7. The constraint in XHSTT instance as shown in the right of Figure 7.7 consists of *Id*, *name*, *required* (true is for hard and false is for soft), *cost function* (the type of cost function), "*AppliesTo"* (the specific resources such as "*BATeacher01*"), *TimeGroups* or *Times* (the specific set of durations or one duration), *minimum* and *maximum* of duration. The constraint contains 5 hours of *minimum* and *maximum* durations. Based on the STP timetable on the left of Figure 7.7, the teacher (*BATeacher01*) should teach 5 slots on Tuesday. The Distribute Split Events constraint was produced based on Figure 7.8 where *5DINAMIK_BA_*2 events should be assigned

on 2 slots per week. The difference between Figure 7.7 and Figure 7.8, is that Figure 7.7 represents the soft constraint of the teacher while Figure 7.8 represents the hard constraints of the event.

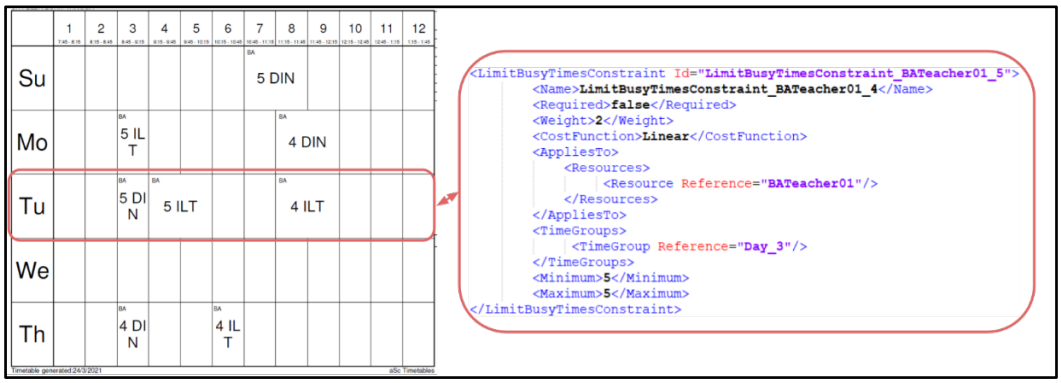


Figure 7.7. Limit Busy Time Constraint

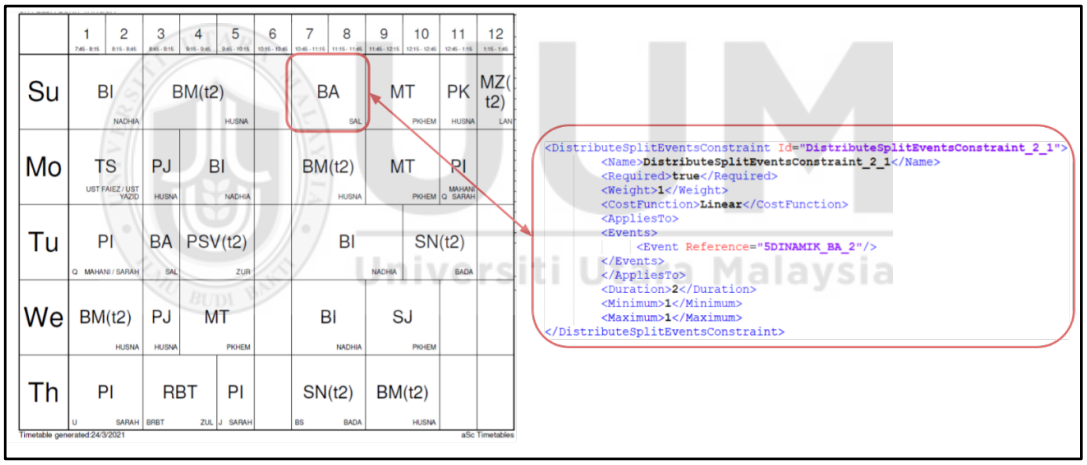


Figure 7.8. Distribute Split Events Constraint

All five components (metadata, times, resources, events, and constraints) stated here were encoded as XHSTT instances in one XML file. A total of over 7000 lines were produced for these five components. The XHSTT instance is divided into three cases based on the complexity of the instance. The first case is 100% of pre-assigned of resources (teacher and class) (as shown in the left part of Figure 7.9) such as "*MTTeacher03*" and "*1ILTZAM*" were pre-assigned in teacher and class resources, respectively. This case is categorized as an easy level named as *MalaysiaSFBT01* data

instance. The second case contains of 100% pre-assigned classes and 70% of pre-assigned teachers. The example of unassigned teachers as shown in the right part of Figure 7.9 has only *ResourceType Reference* of teacher instead of the teacher *Id*. This case is categorized as a normal level named as *MalaysiaSFBT02* data instance. The third case also same with the second case, however only 50% of teachers are pre-assigned. This case is categorized as a hard level named as *MalaysiaSFBT03* data instance. The next section describes on solving the three XHSTT instances.

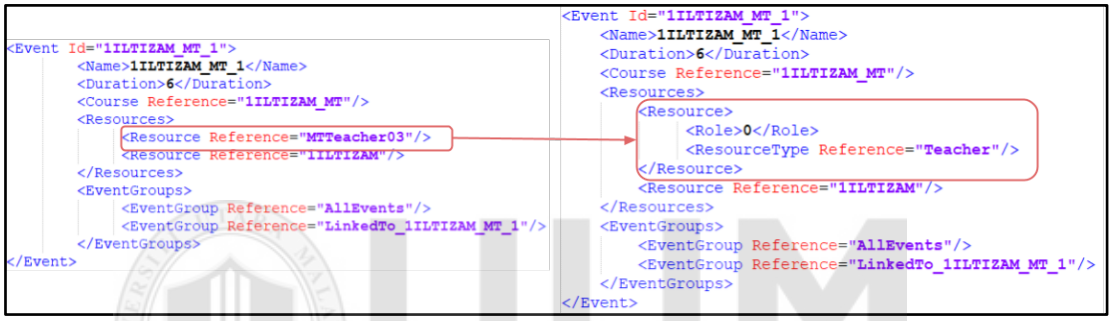


Figure 7.9. Comparison of pre-assigned (right side) and unassigned (left side) teacher

7.3 Experimental and Results

All three XHSTT instances produced earlier were solved using the MCHR construction method to produce the initial solution and improved the solution was improved with MGDA and DHSA methods. The results of the experiments are shown in Table 7.2. Using MCHR, the solution obtained for *MalaysiaSFBT01, MalaysiaSFBT02* and *MalaysiaSFBT03* datasets are 0.00000, 0.00010 and 0.00006, respectively. The worst solution for MalaysiaSFBT02 compared to MalaysiaSFBT03 is influenced by the prioritization of index, demands, and the duration of layer sorting before the time allocation stage in the MCHR model. For MalaysiaSFBT01, a solution value of 0.00000 indicates no violations of hard or soft constraints, making it an optimal solution that

meets the aforementioned eight constraints. The improvement methods which are MGDA and DHSA produces solution of 0.00000 for all datasets.

Table 7.2

Results of The Experiments Based on Difficulty Level Cases

Instance	Difficulty Level Cases	MCHR	MGDA	DHSA	Hybrid DHSA with MGDA
MalaysiaSFBT01	Easy	0.00000	0.00000	0.00000	0.00000
MalaysiaSFBT02	Normal	0.00010	0.00000	0.00000	0.00000
MalaysiaSFBT03	Hard	0.00006	0.00000	0.00000	0.00000

The above results were also validated using HSEval[9] (High School Timetable Evaluator). Figure 7.10 shows the evaluation summary of MalaysiaSFBT01 dataset utilizing the DHSA which satisfy all constraints and produce zero cost. Next, Figure 7.11 shows evaluation summary of MalaysiaSFBT02 dataset applying MGDA which satisfies all constraints and produce zero cost. In Figure 7.12, the evaluation summary of MalaysiaSFBT03 dataset using MCHR shows three Limit Busy Times Constraints with different teachers that were not satisfied which contributed to total of 6 soft constraints. This value is written as 0.00006.

[9] The HSEval High School Timetable Evaluator http://jeffreykingston.id.au/cgi-bin/hseval.cgi

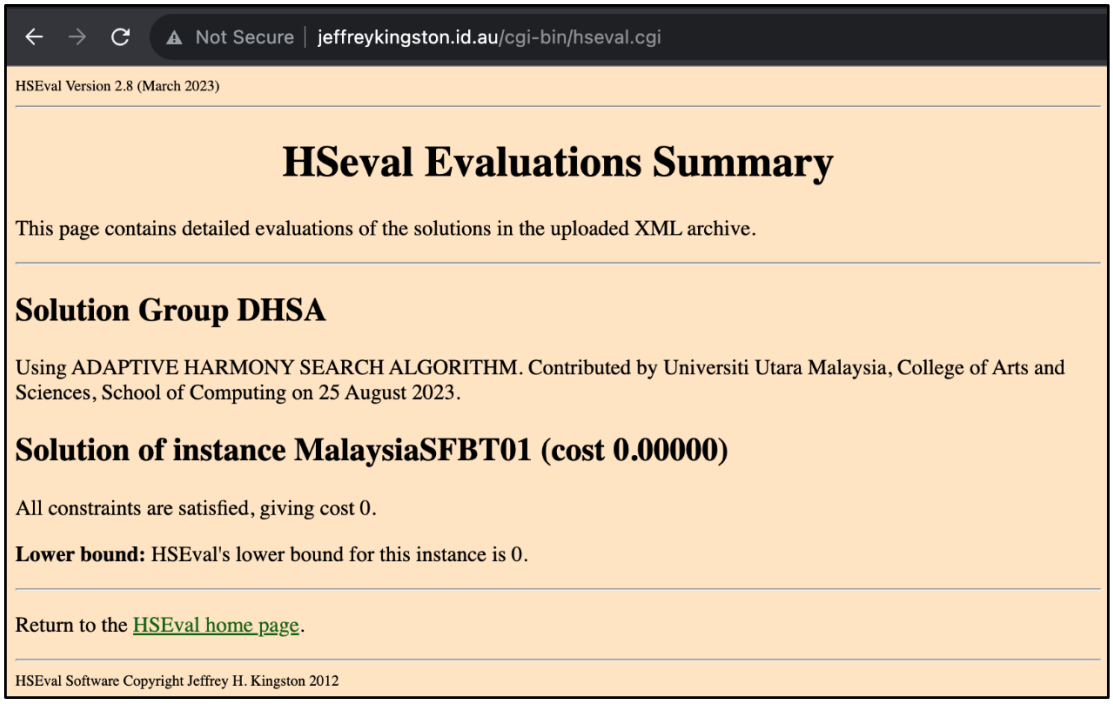


Figure 7.10. HESeval evaluation summary of MalaysiaSFBT01 using DHSA

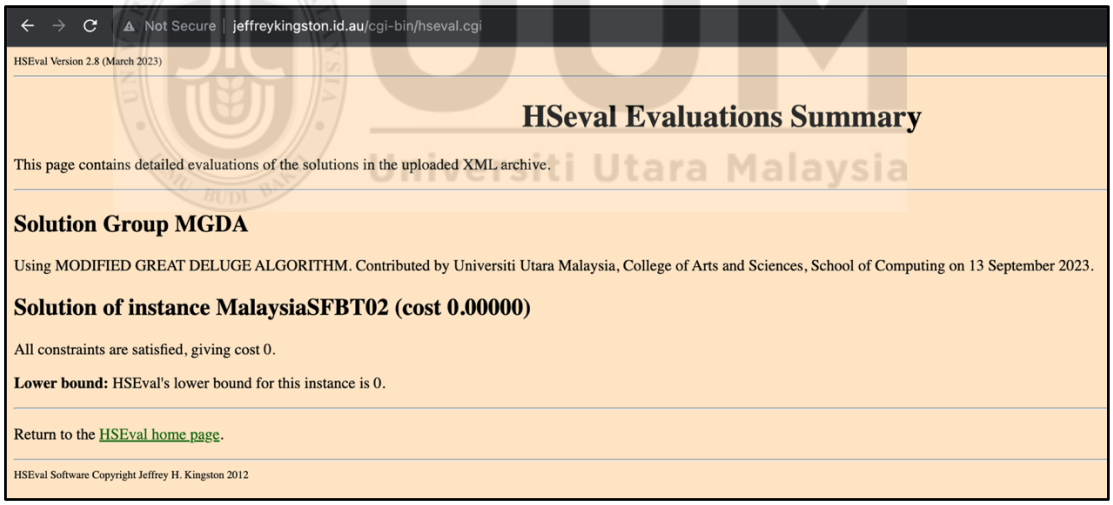
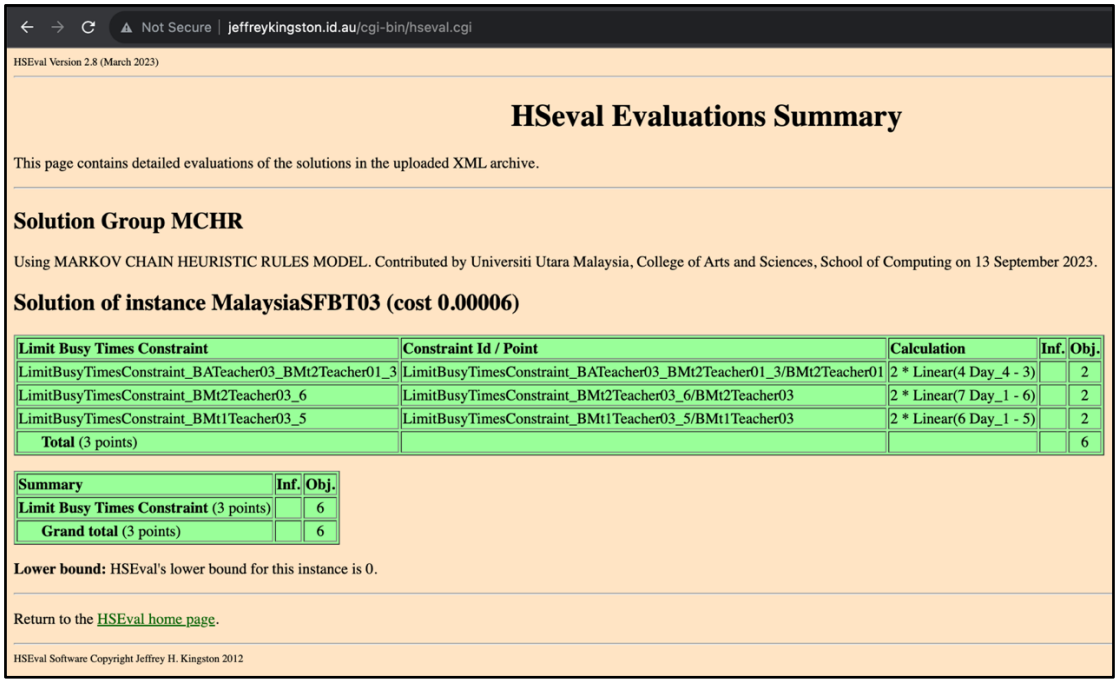


Figure 7.11. HESeval evaluation summary of MalaysiaSFBT02 using MGDA



Not Secure | jeffreykingston.id.au/cgi-bin/hseval.cgi

HSEval Version 2.8 (March 2023)

HSeval Evaluations Summary

This page contains detailed evaluations of the solutions in the uploaded XML archive.

Solution Group MCHR

Using MARKOV CHAIN HEURISTIC RULES MODEL. Contributed by Universiti Utara Malaysia, College of Arts and Sciences, School of Computing on 13 September 2023.

Solution of instance MalaysiaSFBT03 (cost 0.00006)

Limit Busy Times Constraint	Constraint Id / Point	Calculation	Inf.	Obj.
LimitBusyTimesConstraint_BATeacher03_BMt2Teacher01_3	LimitBusyTimesConstraint_BATeacher03_BMt2Teacher01_3/BMt2Teacher01	2 * Linear(4 Day_4 - 3)		2
LimitBusyTimesConstraint_BMt2Teacher03_6	LimitBusyTimesConstraint_BMt2Teacher03_6/BMt2Teacher03	2 * Linear(7 Day_1 - 6)		2
LimitBusyTimesConstraint_BMt1Teacher03_5	LimitBusyTimesConstraint_BMt1Teacher03_5/BMt1Teacher03	2 * Linear(6 Day_1 - 5)		2
Total (3 points)				6

Summary	Inf.	Obj.
Limit Busy Times Constraint (3 points)		6
Grand total (3 points)		6

Lower bound: HSEval's lower bound for this instance is 0.

Return to the HSEval home page.

HSEval Software Copyright Jeffrey H. Kingston 2012

Figure 7.12. HESeval evaluation summary of MalaysiaSFBT03 using MCHR

7.4 Summary

This chapter presents the detail of converting the real-world STP timetable to the XHSTT instance dataset. This study populates the XHSTT real-world instance dataset into three difficulty levels which are easy, normal and hard based on the preassigned of teachers. The real-world data instance has been solved using current construction and improvement methods in this study which produce optimal solution for each difficulty level dataset. The solutions of each difficulty level dataset have been evaluated through HSEval High School Timetable Evaluator. The next Chapter will cover the conclusion of the thesis that consists of general discussion, research contributions, achievements, limitation, and future works.

CHAPTER EIGHT

CONCLUSION AND FUTURE WORK

8.1 Introduction

This chapter highlights the summary of this thesis by on addressing each of the research objective with its theoretical problem, contributions, achievements, limitations and future works.

8.2 Theoretical Problem, Research Contributions and Achievements

The fundamental theoretical problem addressed by this thesis revolves around solving HSTP, which involve balancing multiple hard and soft constraints related to time, resources, and allocation. Existing methods often fell short of effectively optimizing the delicate balance between exploration and exploitation in the search for feasible and optimal timetables. This gap motivated the need for more sophisticated approaches that could address both the theoretical and practical aspects of the problem.

To address this, the primary contribution of the thesis is the hybridization of the DHSA with the MGDA to solve HSTP. This hybrid approach enhances the performance of traditional algorithms by introducing adaptive parameters, neighborhood structures, and decay rate modifications, specifically tailored to the complexities of real-world timetabling. The theoretical gaps that were previously unaddressed, particularly in the handling of parameter linkages and heuristic rule evaluation, have been tackled by integrating the Markov Chain Heuristic Rules Model (MCHR) into the construction process. This model evaluates heuristic rules more effectively to achieve minimum-cost matching in the scheduling process. The thesis' contributions are as follows:

The thesis' primary contribution is the hybridization of adaptive Harmony Search Algorithm with modified Great Deluge Algorithm for solving High School Timetabling Problems through the sub-contributions of construction and improvement algorithms elaborates as follows.

Kingston High School Timetabling Engine (KHE) is a construction algorithm for solving well-known cross-world benchmark instances of high school problems. KHE consists of four stages i.e., structure, time assignment, resource assignment and clean-up stages. In the time assignment stage, the layers generated in the structure stage are ordered in accordance with established criteria, or the so-called heuristic rule. The fundamental factor that ties these heuristic rules, which is the most challenging layer should be allocated first to satisfy the least cost of matching meeting to time assignment. However, the sorting layers is not fulfilled by the heuristic rules used in the current construction KHE algorithms. **In the first contribution**, a new strategy for sorting layers during the time assignment stage of the construction algorithm of KHE is based on the Markov-chain theory, which is used to model the heuristic rules named as Markov Chain Heuristic Rules Model (MCHR). The sorting layers during the time assignment step method seeks to accomplish the minimum-cost matching. The index, duration, and demand of layers are proprieties in the heuristic rules that are used to determine whether the minimum-cost matching strategy is successful. By employing the MCHR, one can evaluate the significance of heuristic rules by correlating different layer attributes like demand, duration, and index. On the bulk of the Extensible Markup Language for High School Timetabling (XHSTT) dataset, the MCHR can generate better results for High School Timetabling Problems (HSTP) solutions when compared

to original construction algorithm of KHE as shown in Chapter Four. The finding of the construction algorithm (population solution for HSTP) in this study can be further expanded using population-based metaheuristics.

The Harmony Search Algorithm (HSA) is an evolutionary algorithm that converts jazz musicians' musical improvisation into a computational method for finding high-quality solutions in optimization problems. Pitch is the term used to describe the aesthetic characteristic that each musical instrument defines, similar to the objective function describing the quality of the choice variables in optimization problems. Numerous earlier studies addressed crucial facets of the evolutionary tactics in adaptive Harmony Search Algorithm (DHSA) such as in the terms of regenerating harmony memory (solution) and adjusting processes (operators). As stated in Chapter Two, the issue with those studies is that they attempted to address the exploration and exploitation issue by concentrating on the interactions between parameters from one perspective while ignoring the iteration position and solution number (two dimensions), behavioral status, parameter linkages for all parameters of HSA. **In the second contribution**, an improved HSA with adaptive parameters by modifying the parameter values based on the iteration position and solution number, behavioral status, and parameter linkages to determine whether to apply various kinds of neighborhood structures to HSTP. Numerous experiments are carried out to evaluate the proposed algorithm and support conclusions about the success of the approach in finding excellent solutions. DHSA is ranked as the best when compared to the best-known metaheuristics results based on average ranking, and it is discovered to be highly competitive with other state-of-the-art methods and able to produce some best solutions. The finding of the DHSA in this

study can be more comprehensively explored by hybridizing with local-based metaheuristics.

The Great Deluge Algorithm (GDA) analogy typically depicts a person climbing a hill and attempting to walk in any direction that avoids getting wet in the hopes of finding a path as the water level rises. GDA decay rate is a word for the avoidance of getting wet that is offered in a computational technique for locating effective solution to optimization problems. According to Chapter Two, particularly for challenging and complicated problems, the feasibility of a solution and its objective function are still separated from the decay rate, which restricts the search to converge to a better solution. **In the third contribution**, a modified version of the GDA (MGDA) is used to solve the HTSP with different neighborhood structures. In the MGDA, two decay rates are suggested to individually reduce the hard and soft constraints. The efficiency of the suggested strategy in avoiding local optimums by using multiple paths of solution was tested using both decay rates. The suggested verification method is effective, according to computational investigations on well-known benchmark examples of HSTP. The proposed strategy is ranked as the fourth best among state-of-the-art approaches and the second best among metaheuristic approaches. The proposed of MGDA can be hybridized with the second contribution which is DHSA.

The previous hybridization of HSA with GDA is widely used in the perspective of in which the parameters of HSA are not handled by GDA and vice versa. In other words, the jazz musicians' improvisational music has a little bit of a faulty pitch, and the musician might notice a new trajectory of adjusting based on the faulty pitch. Regarding this claim, the pitch of the HSA has been changed depending on the GDA decay rate

threshold. **In the fourth contribution**, the integration of DHSA and MGDA is enhanced with an added component: the DHSA pitch adjustment, which is influenced by the water level of MGDA concerning behavioral neighborhood movements. When comparing the average ranks with the state-of-the-art methods, this combination of DHSA and MGDA delivered outstanding solution quality, securing a third-place ranking.

The use of a standardized data format or language to identify HSTP has been studied by researchers. XHSTT describes the actual problem that exists in several nations worldwide e.g., USA, Italy, Brazil, Australia, and Spain. The main emphasis of earlier works focused more on the methodologies on solving HSTP rather than the specifics on how to generate real-world datasets for HSTP. From this point of view, the step by step of producing Malaysian real-world HSTP datasets based on the XHSTT format is illustrated. **In the fifth contribution**, it is a kind of practical contribution that offers details on producing Malaysian real-world datasets of HSTP based on the dataset format for the XHSTT. HSTP timetable of 2021 was obtained from the primary school at Bukit Kayu Hitam, Kedah, Malaysia. The constraints of the Malaysian real-world case problem are identical with the XHSTT benchmark datasets, and they are divided into three preassigned teachers percentage numbers that represented the difficulty levels of easy, normal and hard. For all the difficulty level based on preassigned teachers, the real-world dataset managed to be optimally solved using both proposed construction and improvement. In addition, the real-world XHSTT data instance can be tested by another researcher.

In summary, the investigation research has carefully progressed through a number of objectives that were determined at the beginning. The first objective, which was to optimize the time assignment phase of the KHE algorithm, was successfully accomplished by use of the MCHR and was elaborated upon in Chapter Four. The second objective was on optimizing the Harmony Search Algorithm, which led to the development of the DHSA, which is covered in Chapter Five. Subsequently, the study explored the limitations of the standard GDA, resulting in a modification of the GDA (MGDA), whose features and benefits are discussed in Chapter Six. The fourth objective combined the best features of DHSA and MGDA to create a powerful hybrid approach which the details are covered in Chapter Six. Table 8.1 provides an overview of the accomplishments linked to each objective.

Table 8.1

Summary of Research Objectives with Related Chapters

Contribution Number	Research Objective Achieved	Chapter
1	Enhancement of sorting layers in KHE using MCHR	Four
2	Refinement of HSA through DHSA	Five
3	Addressing GDA limitations via MGDA	Six
4	Integration of DHSA & MGDA	Six

8.3 Future Work

Currently, the MCHR model does not incorporate the self-adaptive concept into its properties, namely demand, duration, and index. An adaptive version of the MCHR, which is based on additional criteria, can intelligently modify the priority value of layers within the MCHR to enhance the quality of solutions. These additional criteria involve setting β and α as random numbers in the MCHR model during the layer sorting stage, which are currently set at 0.99.

At present, neighborhood structures in improvement methods (such as DHSA, MGDA, and the hybridization of DHSA and MGDA) are selected at random. Incorporating different topologies, such as star, ring, and wheel neighborhood structures, could enhance the performance of these improvement methods.

Moreover, the MGDA has lack of NS diversity sequence (management) which increase the stagnation of local search. Gradient descent (such as variable neighborhood descent) with hybridized DHSA and MGDA will perhaps provide a way to minimize the stagnation of local search. Gradient descent includes a variety of NS sequences, such as rearranging the NS order to enhance solutions within a particular search space.

Besides, there is still a lack of details about how to generate constraints of Malaysian real-world datasets of HSTP based on the dataset format for the XHSTT. Additionally, there is no web application for formatting the components (metadata, times, resources, events, and constraints) based Malaysian real-world problem. Future development will involve creating a prototype that will help to generate constraints and administer Malaysian real-world datasets with ease.

REFERENCES

Abdulkhaleq, M. T., Rashid, T. A., Alsadoon, A., Hassan, B. A., Mohammadi, M., Abdullah, J. M., Chhabra, A., Ali, S. L., Othman, R. N., Hasan, H. A., Azad, S., Mahmood, N. A., Abdalrahman, S. S., Rasul, H. O., Bacanin, N., & Vimal, S. (2022). Harmony search: Current studies and uses on healthcare systems. *Artificial Intelligence in Medicine*, *131*(i), 102348. https://doi.org/10.1016/j.artmed.2022.102348

Abramson, D. (1991). Constructing school timetables using simulated annealing. Sequential and parallel algorithms. *Management Science*, *37*(1), 98–113. https://doi.org/10.1287/mnsc.37.1.98

Abu Doush, I., Al-Betar, M. A., Awadallah, M. A., Alyasseri, Z. A. A., Makhadmeh, S. N., & El-Abd, M. (2022). Island neighboring heuristics harmony search algorithm for flow shop scheduling with blocking. *Swarm and Evolutionary Computation*, *74*, 101127. https://doi.org/10.1016/j.swevo.2022.101127

Abualigah, L., Diabat, A., & Geem, Z. W. (2020). A Comprehensive Survey of the Harmony Search Algorithm in Clustering Applications. *Applied Sciences*, *10*(11), 3827. https://doi.org/10.3390/app10113827

Acan, A., & Ünveren, A. (2015). A great deluge and tabu search hybrid with two-stage memory support for quadratic assignment problem. *Applied Soft Computing Journal*, *36*, 185–203. https://doi.org/10.1016/j.asoc.2015.06.061

Ahmed, L. N., Özcan, E., & Kheiri, A. (2015). Solving high school timetabling problems worldwide using selection hyper-heuristics. *Expert Systems with Applications*, *42*(13), 5463–5471. https://doi.org/10.1016/j.eswa.2015.02.059

Aït-Sahalia, Y., & Hansen, L. P. (2010). *Handbook of Financial Econometrics: Applications*. Elsevier. https://doi.org/10.1016/C2009-0-62219-4

Al-Betar, M. A., Ahmad, O. N., Khader, A. T., & Awadallah, M. A. (2013). Incorporating Great Deluge with Harmony Search for Global Optimization Problems. In *Advances in Intelligent Systems and Computing: Vol. 201 AISC* (Issue VOL. 1, pp. 275–286). https://doi.org/10.1007/978-81-322-1038-2_24

Al-Betar, M. A., & Khader, A. T. (2012). A harmony search algorithm for university course timetabling. *Annals of Operations Research*, *194*(1), 3–31. https://doi.org/10.1007/s10479-010-0769-z

Al-Betar, M. A., Khader, A. T., & Zaman, M. (2012). University course timetabling

using a hybrid harmony search metaheuristic algorithm. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, *42*(5), 664–681. https://doi.org/10.1109/TSMCC.2011.2174356

Andrade, P. R. de L., Steiner, M. T. A., & Góes, A. R. T. (2019). Optimization in timetabling in schools using a mathematical model, local search and Iterated Local Search procedures. *Gestão & Produção*, *26*(4), 1–23. https://doi.org/10.1590/0104-530x3241-19

Angadi, B. M., Kakkasageri, M. S., & Manvi, S. S. (2021). Computational intelligence techniques for localization and clustering in wireless sensor networks. In *Recent Trends in Computational Intelligence Enabled Research* (pp. 23–40). Elsevier. https://doi.org/10.1016/B978-0-12-822844-9.00011-6

Anwar, K., Khader, A. T., Al-Betar, M. A., & Awadallah, M. A. (2013). Harmony Search-based Hyper-heuristic for examination timetabling. *2013 IEEE 9th International Colloquium on Signal Processing and Its Applications*, *March*, 176–181. https://doi.org/10.1109/CSPA.2013.6530037

Apt, K. (2003). Principles of Constraint Programming. In *Principles of Constraint Programming*. Cambridge University Press. https://doi.org/10.1017/CBO9780511615320

Askarzadeh, A., & Rashedi, E. (2017). *Harmony Search Algorithm* (pp. 1–36). https://doi.org/10.4018/978-1-5225-2322-2.ch001

Assad, A., & Deep, K. (2018). A Hybrid Harmony search and Simulated Annealing algorithm for continuous optimization. *Information Sciences*, *450*, 246–266. https://doi.org/10.1016/j.ins.2018.03.042

Awadallah, M. A., Al-Betar, M. A., Khader, A. T., Bolaji, A. L., & Alkoffash, M. (2017). Hybridization of harmony search with hill climbing for highly constrained nurse rostering problem. *Neural Computing and Applications*, *28*(3), 463–482. https://doi.org/10.1007/s00521-015-2076-8

Babaei, H., Karimpour, J., & Hadidi, A. (2015). A survey of approaches for university course timetabling problem. *Computers & Industrial Engineering*, *86*, 43–59. https://doi.org/10.1016/j.cie.2014.11.010

Bashab, A., Ibrahim, A. O., AbedElgabar, E. E., Ismail, M. A., Elsafi, A., Ahmed, A., & Abraham, A. (2020). A systematic mapping study on solving university timetabling problems using meta-heuristic algorithms. *Neural Computing and Applications*, *32*(23), 17397–17432. https://doi.org/10.1007/s00521-020-05110-3

Baykasoglu, A. (2012). Design optimization with chaos embedded great deluge algorithm. *Applied Soft Computing*, *12*(3), 1055–1067. https://doi.org/10.1016/j.asoc.2011.11.018

Bazaraa, M. S., & Shetty, C. M. (1979). *Nonlinear Programming: Theory and Algorithms*. Wiley. https://books.google.co.th/books?id=fGpRAAAAMAAJ

Berghuis, J., van der Heiden, A. J., & Bakker, R. (1964). The preparation of school time tables by electronic computer. *BIT Numerical Mathematics*, *4*(2), 106–114. https://doi.org/10.1007/BF01939852

Birbas, T., Daskalaki, S., & Housos, E. (2009). School timetabling for quality student and teacher schedules. *Journal of Scheduling*, *12*(2), 177–197. https://doi.org/10.1007/s10951-008-0088-2

Bishop, C. M. (2006). Pattern recognition and machine learning. *Springer Google Schola*, *2*, 5–43.

Blum, C., & Roli, A. (2003). Metaheuristics in combinatorial optimization. *ACM Computing Surveys*, *35*(3), 268–308. https://doi.org/10.1145/937503.937505

Boland, N., Hughes, B. D., Merlot, L. T. G., & Stuckey, P. J. (2008). New integer linear programming approaches for course timetabling. *Computers and Operations Research*, *35*(7), 2209–2233. https://doi.org/10.1016/j.cor.2006.10.016

Braak, M. ter. (2012). *A Hyperheuristic for generating Timetables in the XHSTT Format*.

Brahim, B., Kobayashi, M., Al Ali, M., Khatir, T., & Elaissaoui Elmeliani, M. E. A. (2024). Metaheuristic Optimization Algorithms: an overview. *HCMCOU Journal of Science – Advances in Computational Structures*. https://doi.org/10.46223/HCMCOUJS.acs.en.14.1.47.2024

Brambila-Hernández, J. A., García-Morales, M. Á., Fraire-Huacuja, H. J., Villegas-Huerta, E., & Becerra-del-Ángel, A. (2023). Hybrid Harmony Search Optimization Algorithm for Continuous Functions. *Mathematical and Computational Applications*, *28*(2), 29. https://doi.org/10.3390/mca28020029

Brilliant.org. (2020). *Markov Chains*. Brilliant.Org. https://brilliant.org/wiki/markov-chains/

Brito, S. S., Fonseca, G. H. G., Toffolo, T. A. M., Santos, H. G., & Souza, M. J. F. (2012). A SA-VNS approach for the High School Timetabling Problem. *Electronic Notes in Discrete Mathematics*, *39*, 169–176.

https://doi.org/10.1016/j.endm.2012.10.023

Burke, E. K., Bykov, Y., & Burke, C. C. (2006). Solving Exam Timetabling Problems with the Flex-Deluge Algorithm. *In Proceedings of the 6th International Conference Practice and Theory of Automated Timetabling VI (PATAT 2006), Brno, Czech Republic, 30 August–1 September 2006; Pp. 370–372. 62.*, 9–11.

Burke, E. K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., & Qu, R. (2013). Hyper-heuristics: a survey of the state of the art. *Journal of the Operational Research Society*, *64*(12), 1695–1724. https://doi.org/10.1057/jors.2013.71

Ceschia, S., Di Gaspero, L., & Schaerf, A. (2022). Educational Timetabling: Problems, Benchmarks, and State-of-the-Art Results. *ArXiv*. https://doi.org/10.48550/ARXIV.2201.07525

Ceschia, S., Di Gaspero, L., & Schaerf, A. (2023). Educational timetabling: Problems, benchmarks, and state-of-the-art results. *European Journal of Operational Research*, *308*(1), 1–18. https://doi.org/10.1016/j.ejor.2022.07.011

Chen, C., Heath, R. W., Bovik, A. C., & de Veciana, G. (2013). A Markov Decision Model for Adaptive Scheduling of Stored Scalable Videos. *IEEE Transactions on Circuits and Systems for Video Technology*, *23*(6), 1081–1095. https://doi.org/10.1109/TCSVT.2013.2254896

Chen, J., Pan, Q., & Li, J. (2012). Harmony search algorithm with dynamic control parameters. *Applied Mathematics and Computation*, *219*(2), 592–604. https://doi.org/10.1016/j.amc.2012.06.048

Choudhuri, R., & Ravi, V. (2010). *A Hybrid Harmony Search and Modified Great Deluge Algorithm for Unconstrained Optimisation*. https://idrbt.ac.in/assets/alumni/PT-2010/Ravidutt_2010.pdf

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). *Introduction to algorithms*. MIT press.

Cuevas, E., Diaz, P., & Camarena, O. (2021). *Experimental Analysis Between Exploration and Exploitation* (pp. 249–269). https://doi.org/10.1007/978-3-030-58100-8_10

Cuevas, E., Luque, A., Morales Castañeda, B., & Rivera, B. (2024). *A Measure of Diversity for Metaheuristic Algorithms Employing Population-Based Approaches* (pp. 49–72). https://doi.org/10.1007/978-3-031-63053-8_3

Cuevas, E., Zaldívar, D., & Pérez-Cisneros, M. (2024). *Introduction to Metaheuristic*

Schemes: Characteristics, Properties, and Importance in Solving Optimization Problems (pp. 1–9). https://doi.org/10.1007/978-3-031-45561-2_1

da Fonseca, G. H. G., Santos, H. G., Toffolo, T. Â. M., Brito, S. S., & Souza, M. J. F. (2016a). GOAL solver: a hybrid local search based solver for high school timetabling. *Annals of Operations Research*, *239*(1), 77–97. https://doi.org/10.1007/s10479-014-1685-4

da Fonseca, G. H. G., Santos, H. G., Toffolo, T. Â. M., Brito, S. S., & Souza, M. J. F. (2016b). GOAL solver: a hybrid local search based solver for high school timetabling. *Annals of Operations Research*, *239*(1), 77–97. https://doi.org/10.1007/s10479-014-1685-4

de Souza Alencar, W., Abdala Rfaei Jradi, W., Dantas do Nascimento, H. A., Felix, J. P., & Alves de Melo Nunes Soares, F. A. (2020). An Empirical Methodological Study of Evaluation Methods Applied to Educational Timetabling Visualizations. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Vol. 12509 LNCS* (pp. 209–223). https://doi.org/10.1007/978-3-030-64556-4_17

Deb, K. (1999). An introduction to genetic algorithms. *Sadhana*, *24*(4–5), 293–315. https://doi.org/10.1007/BF02823145

Demirović, E., & Musliu, N. (2017). MaxSAT-based large neighborhood search for high school timetabling. *Computers & Operations Research*, *78*, 172–180. https://doi.org/10.1016/j.cor.2016.08.004

Demirović, E., & Stuckey, P. J. (2018). *Constraint Programming for High School Timetabling: A Scheduling-Based Model with Hot Starts* (W.-J. van Hoeve (ed.); Vol. 10848, pp. 135–152). Springer International Publishing. https://doi.org/10.1007/978-3-319-93031-2_10

Domrös, J., & Homberger, J. (2012). An evolutionary algorithm for high school timetabling. *PATAT 2012 - Proceedings of the 9th International Conference on the Practice and Theory of Automated Timetabling*, *August*, 485–488.

Dorneles, Á. P., de Araújo, O. C. B., & Buriol, L. S. (2014). A fix-and-optimize heuristic for the high school timetabling problem. *Computers & Operations Research*, *52*, 29–38. https://doi.org/10.1016/j.cor.2014.06.023

Du, K.-L., & Swamy, M. N. S. (2016). Search and Optimization by Metaheuristics. In *Search and Optimization by Metaheuristics*. Springer International Publishing. https://doi.org/10.1007/978-3-319-41192-7

Dubey, M., Kumar, V., Kaur, M., & Dao, T.-P. (2021). A Systematic Review on Harmony Search Algorithm: Theory, Literature, and Applications. *Mathematical Problems in Engineering*, *2021*, 1–22. https://doi.org/10.1155/2021/5594267

Dueck, G. (1993). New Optimization Heuristics. *Journal of Computational Physics*, *104*(1), 86–92. https://doi.org/10.1006/jcph.1993.1010

Edrisi, A., Bagherzadeh, K., & Nadi, A. (2019). Applying Markov decision process to adaptive dynamic route selection model. *Proceedings of the Institution of Civil Engineers - Transport*, 1–14. https://doi.org/10.1680/jtran.19.00085

Eng, K. L., Muhammed, A., Mohamed, M. A., & Hasan, S. (2020). A hybrid heuristic of Variable Neighbourhood Descent and Great Deluge algorithm for efficient task scheduling in Grid computing. *European Journal of Operational Research*, *284*(1), 75–86. https://doi.org/10.1016/j.ejor.2019.12.006

Eng, K., Muhammed, A., Mohamed, M. A., & Hasan, S. (2020). A hybrid heuristic of Variable Neighbourhood Descent and Great Deluge algorithm for efficient task scheduling in Grid computing. *European Journal of Operational Research*, *284*(1), 75–86. https://doi.org/10.1016/j.ejor.2019.12.006

Even, S., Itai, A., & Shamir, A. (1976). On the Complexity of Timetable and Multicommodity Flow Problems. *SIAM Journal on Computing*, *5*(4), 691–703. https://doi.org/10.1137/0205048

Feng, Z.-Y., Guo, H., Liu, Z.-T., Xu, L., & She, J. (2017). Hybridization of harmony search with Nelder-Mead algorithm for combined heat and power economic dispatch problem. *2017 36th Chinese Control Conference (CCC)*, 2790–2795. https://doi.org/10.23919/ChiCC.2017.8027787

Fernandes, C., Caldeira, J. P., Melicio, F., & Rosa, A. (1999). High school weekly timetabling by evolutionary algorithms. *Proceedings of the 1999 ACM Symposium on Applied Computing*, 344–350. https://doi.org/10.1145/298151.298379

Fesanghary, M., Mahdavi, M., Minary-Jolandan, M., & Alizadeh, Y. (2008). Hybridizing harmony search algorithm with sequential quadratic programming for engineering optimization problems. *Computer Methods in Applied Mechanics and Engineering*, *197*(33–40), 3080–3091. https://doi.org/10.1016/j.cma.2008.02.006

Filho, G. R., & Nogueira Lorena, L. A. (2001). A constructive evolutionary approach to school timetabling. *Lecture Notes in Computer Science*, *2037*(LNCS), 130–

139. https://doi.org/10.1007/3-540-45365-2_14

Fonseca, G. H. G., Brito, S. S., & Santos, H. G. (2012). A Simulated Annealing Based Approach to the High School Timetabling Problem. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Vol. 7435 LNCS* (pp. 540–549). https://doi.org/10.1007/978-3-642-32639-4_66

Fonseca, G. H. G., & Santos, H. G. (2014). Variable Neighborhood Search based algorithms for high school timetabling. *Computers & Operations Research*, *52*(June 2020), 203–208. https://doi.org/10.1016/j.cor.2013.11.012

Fonseca, G. H. G., Santos, H. G., & Carrano, E. G. (2016a). Late acceptance hill-climbing for high school timetabling. *Journal of Scheduling*, *19*(4), 453–465. https://doi.org/10.1007/s10951-015-0458-5

Fonseca, G. H. G., Santos, H. G., & Carrano, E. G. (2016b). Integrating matheuristics and metaheuristics for timetabling. *Computers & Operations Research*, *74*, 108–117. https://doi.org/10.1016/j.cor.2016.04.016

Fonseca, G. H. G., Santos, H. G., & Carrano, E. G. (2016c). Integrating matheuristics and metaheuristics for timetabling. *Computers & Operations Research*, *74*, 108–117. https://doi.org/10.1016/j.cor.2016.04.016

Fonseca, G. H. G., Santos, H. G., Carrano, E. G., & Stidsen, T. J. R. (2017). Integer programming techniques for educational timetabling. *European Journal of Operational Research*, *262*(1), 28–39. https://doi.org/10.1016/j.ejor.2017.03.020

Fonseca, G. H. G., Santos, H. G., Toffolo, T. A. M., Brito, S. S., & Souza, M. J. F. (2012). A sa-ils approach for the high school timetabling problem. *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling ({PATAT} 2012)*, *August*, 493–496. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.308.4488&rep=rep1&type=pdf#page=494

Gashgari, R., Alhashimi, L., Obaid, R., Palaniswamy, T., Aljawi, L., & Alamoudi, A. (2018). A Survey on Exam Scheduling Techniques. *2018 1st International Conference on Computer Applications & Information Security (ICCAIS)*, *November*, 1–5. https://doi.org/10.1109/CAIS.2018.8441950

Gashgari, R., Alhashimi, L., Obaid, R., Palaniswamy, T., Aljawi, L., & Alamoudi, A. (2021). A Survey of Computational Intelligence in Educational Timetabling. *International Journal of Machine Learning and Computing*, *11*(1), 40–47.

https://doi.org/10.18178/ijmlc.2021.11.1.1012

Geem, Z. W. (2010). State-of-the-Art in the Structure of Harmony Search Algorithm. In *Recent Advances In Harmony Search Algorithm* (Vol. 270, pp. 1–10). https://doi.org/10.1007/978-3-642-04317-8_1

Geem, Z. W., Kim, J. H., & Loganathan, G. V. (2001). A New Heuristic Optimization Algorithm: Harmony Search. *SIMULATION*, *76*(2), 60–68. https://doi.org/10.1177/003754970107600201

Gheisarnejad, M. (2018). An effective hybrid harmony search and cuckoo optimization algorithm based fuzzy PID controller for load frequency control. *Applied Soft Computing*, *65*, 121–138. https://doi.org/10.1016/j.asoc.2018.01.007

Glover, F., & Laguna, M. (1997). Tabu Search Principles. In *Tabu Search* (pp. 125–151). Springer US. https://doi.org/10.1007/978-1-4615-6089-0_5

Godim Da Fonseca, G. H. (2017). *Formulations and Algorithms for Timetabling*. *April*. https://www.ppgee.ufmg.br/defesas/1351D.PDF

Gomes, A. C. L., Ravetti, M. G., & Carrano, E. G. (2020). Multi-objective matheuristic for minimization of total tardiness and energy costs in a steel industry heat treatment line. *Computers & Industrial Engineering*, 106929. https://doi.org/10.1016/j.cie.2020.106929

Goode, E., Syme, S., & Nieuwoudt, J. E. (2024). The impact of immersive scheduling on student learning and success in an Australian pathways program. *Innovations in Education and Teaching International*, *61*(2), 263–275. https://doi.org/10.1080/14703297.2022.2157304

Guha, R., Ghosh, M., Kapri, S., Shaw, S., Mutsuddi, S., Bhateja, V., & Sarkar, R. (2021). Deluge based Genetic Algorithm for feature selection. *Evolutionary Intelligence*, *14*(2), 357–367. https://doi.org/10.1007/s12065-019-00218-5

Hamadi, Y., & Saïs, L. (2018). Handbook of Parallel Constraint Reasoning. In Y. Hamadi & L. Sais (Eds.), *Handbook of Parallel Constraint Reasoning*. Springer International Publishing. https://doi.org/10.1007/978-3-319-63516-3

Hao, X., Liu, J., Zhang, Y., & Sanga, G. (2021). Mathematical model and simulated annealing algorithm for Chinese high school timetabling problems under the new curriculum innovation. *Frontiers of Computer Science*, *15*(1), 151309. https://doi.org/10.1007/s11704-020-9102-4

Hartmanis, J., & Leeuwen, J. Van. (2001). *Applications of Evolutionary Computing*

(E. J. W. Boers (ed.); Vol. 2037). Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-45365-2

Hasanipanah, M., Keshtegar, B., Thai, D.-K., & Troung, N.-T. (2022). An ANN-adaptive dynamical harmony search algorithm to approximate the flyrock resulting from blasting. *Engineering with Computers*, *38*(2), 1257–1269. https://doi.org/10.1007/s00366-020-01105-9

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770–778.

Hooshmand, S., Behshameh, M., & Hamidi, O. (2013). A Tabu Search Algorithm With Efficient Diversification Strategy for High School Timetabling Problem. *International Journal of Computer Science and Information Technology*, *5*(4), 21–34. https://doi.org/10.5121/ijcsit.2013.5402

Huang, C.-Y. (Ric), Lai, C.-Y., & Cheng, K.-T. (Tim). (2009). Fundamentals of algorithms. In *Electronic Design Automation* (pp. 173–234). Elsevier. https://doi.org/10.1016/B978-0-12-374364-0.50011-4

Hussain, K., Salleh, M. N. M., Cheng, S., & Shi, Y. (2019). On the exploration and exploitation in popular swarm-based metaheuristic algorithms. *Neural Computing and Applications*, *31*(11), 7665–7683. https://doi.org/10.1007/s00521-018-3592-0

Innocente, M. S., & Sienz, J. (2021). Constraint-Handling Techniques for Particle Swarm Optimization Algorithms. *CoRR*, *abs/2101.1*. https://doi.org/10.48550/arXiv.2101.10933

Jaddi, N. S., & Abdullah, S. (2013). Hybrid of genetic algorithm and great deluge algorithm for rough set attribute reduction. *TURKISH JOURNAL OF ELECTRICAL ENGINEERING & COMPUTER SCIENCES*, *21*(6), 1737–1750. https://doi.org/10.3906/elk-1202-113

Jiang, J., & Fan, J. A. (2023). *Large-scale global optimization of ultra-high dimensional non-convex landscapes based on generative neural networks*. http://arxiv.org/abs/2307.04065

Jonathan, R. (2019). *Improving the ILS-TQ technique for The High School Timetabling Problem*. Agrarian University of Ecuador.

Jordan, M. I., & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, *349*(6245), 255–260.

Junn, K. Y., Obit, J. H., & Alfred, R. (2017). Comparison of Simulated Annealing and Great Deluge Algorithms for University Course Timetabling Problems (UCTP). *Advanced Science Letters*, *23*(11), 11413–11417. https://doi.org/10.1166/asl.2017.10295

Kamrani, M., Srinivasan, A. R., Chakraborty, S., & Khattak, A. J. (2020). Applying Markov decision process to understand driving decisions using basic safety messages data. *Transportation Research Part C: Emerging Technologies*, *115*(February), 102642. https://doi.org/10.1016/j.trc.2020.102642

Karras, C., Karras, A., Avlonitis, M., & Sioutas, S. (2022). *An Overview of MCMC Methods: From Theory to Applications* (pp. 319–332). https://doi.org/10.1007/978-3-031-08341-9_26

Katsaragakis, I., Tassopoulos, I., & Beligiannis, G. (2015). A Comparative Study of Modern Heuristics on the School Timetabling Problem. *Algorithms*, *8*(3), 723–742. https://doi.org/10.3390/a8030723

Khan, M. W., Khanam, F., Khan, N. S. M., Ahmed, Z. A., & Usman, M. (2018). Towards Global Peace & Spiritual Living. In *Spirit of Islam* (Issue 72).

Kheiri, A., & Keedwell, E. (2017). A Hidden Markov Model Approach to the Problem of Heuristic Selection in Hyper-Heuristics with a Case Study in High School Timetabling Problems. *Evolutionary Computation*, *25*(3), 473–501. https://doi.org/10.1162/evco_a_00186

Kheiri, A., Özcan, E., & Parkes, A. J. (2016). A stochastic local search algorithm with adaptive acceptance for high-school timetabling. *Annals of Operations Research*, *239*(1), 135–151. https://doi.org/10.1007/s10479-014-1660-0

Kifah, S., & Abdullah, S. (2015). An adaptive non-linear great deluge algorithm for the patient-admission problem. *Information Sciences*, *295*, 573–585. https://doi.org/10.1016/j.ins.2014.10.004

Kingston, J. (2022). *A Software Library for High School Timetabling and Nurse Rostering*. http://jeffreykingston.id.au/khe/

Kingston, J. H. (2005). A Tiling Algorithm for High School Timetabling. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Vol. 3616 LNCS* (pp. 208–225). https://doi.org/10.1007/11593577_13

Kingston, J. H. (2007). Hierarchical Timetable Construction. In E. K. Burke & H. Rudová (Eds.), *Practice and Theory of Automated Timetabling VI* (pp. 294–307).

Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-77345-0_19

Kingston, J. H. (2012). Resource assignment in high school timetabling. *Annals of Operations Research*, *194*(1), 241–254. https://doi.org/10.1007/s10479-010-0695-0

Kingston, J. H. (2014). KHE14: An Algorithm for High School Timetabling. *10th International Conference of the Practice and Theory of Automated Timetabling (PATAT 2014)*, 26–29.

Kingston, J. H. (2018). *High School Timetable Data Format Specification: Glossary*. http://jeffreykingston.id.au/cgi-bin/hseval.cgi?op=spec&part=glossary

Kingston, J. H. (2024). KHE24 : Towards a Practical Solver for Nurse Rostering (Abstract). *PATAT 2024, The 14th International Conference on Practice and Theory of Automated Timetabling, Copenhagen, Denmark*, 1–16.

Kitchin, R. (2014). *The data revolution: Big data, open data, data infrastructures and their consequences*. Sage.

Koopialipoor, M., & Noorbakhsh, A. (2020a). Applications of Artificial Intelligence Techniques in Optimizing Drilling. In *Emerging Trends in Mechatronics*. IntechOpen. https://doi.org/10.5772/intechopen.85398

Koopialipoor, M., & Noorbakhsh, A. (2020b). Applications of Artificial Intelligence Techniques in Optimizing Drilling. In *Emerging Trends in Mechatronics*. IntechOpen. https://doi.org/10.5772/intechopen.85398

Kristiansen, S., Sørensen, M., & Stidsen, T. R. (2015a). Integer programming for the generalized high school timetabling problem. *Journal of Scheduling*, *18*(4), 377–392. https://doi.org/10.1007/s10951-014-0405-x

Kristiansen, S., Sørensen, M., & Stidsen, T. R. (2015b). Integer programming for the generalized high school timetabling problem. *Journal of Scheduling*, *18*(4), 377–392. https://doi.org/10.1007/s10951-014-0405-x

Kristiansen, S., & Stidsen, T. R. (2012). Adaptive large neighborhood search for student sectioning at Danish high schools. *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, *August*, 29–31.

Landa-Silva, D., & Obit, J. H. (2008a). Great deluge with non-linear decay rate for solving course timetabling problems. *2008 4th International IEEE Conference Intelligent Systems, IS 2008*, *1*, 811–818. https://doi.org/10.1109/IS.2008.4670447

Landa-Silva, D., & Obit, J. H. (2008b). Great deluge with non-linear decay rate for solving course timetabling problems. *2008 4th International IEEE Conference Intelligent Systems*, *1*(October 2008), 8-11-8–18. https://doi.org/10.1109/IS.2008.4670447

Lehman, J. (2017). Analyzing deception, evolvability, and behavioral rarity in evolutionary robotics. *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 1479–1486. https://doi.org/10.1145/3067695.3082514

Levin, D., Peres, Y., Wilmer, E. (2008). *Markov Chains and Mixing Times*.

Li, W., Özcan, E., Drake, J. H., & Maashi, M. (2023). A Generality Analysis of Multiobjective Hyper-heuristics. *Information Sciences*. https://doi.org/10.1016/j.ins.2023.01.047

Li, X., Qin, K., Zeng, B., Gao, L., & Wang, L. (2017). A dynamic parameter controlled harmony search algorithm for assembly sequence planning. *The International Journal of Advanced Manufacturing Technology*, *92*(9–12), 3399–3411. https://doi.org/10.1007/s00170-017-0352-8

Liu, M., & Yu, D. (2023). Towards intelligent E-learning systems. *Education and Information Technologies*, *28*(7), 7845–7876. https://doi.org/10.1007/s10639-022-11479-6

Lourenço, H. R., Martin, O. C., & Stützle, T. (2010). *Iterated Local Search: Framework and Applications* (pp. 363–397). https://doi.org/10.1007/978-1-4419-1665-5_12

Ma, Z., Wu, G., Suganthan, P. N., Song, A., & Luo, Q. (2023). Performance assessment and exhaustive listing of 500+ nature-inspired metaheuristic algorithms. *Swarm and Evolutionary Computation*, *77*, 101248. https://doi.org/10.1016/j.swevo.2023.101248

Mafarja, M., & Abdullah, S. (2011). Modified great deluge for attribute reduction in rough set theory. *2011 Eighth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, *3*(July), 1464–1469. https://doi.org/10.1109/FSKD.2011.6019832

Mafarja, M., & Abdullah, S. (2014). Fuzzy Modified Great Deluge Algorithm for Attribute Reduction. In *Advances in Intelligent Systems and Computing* (Vol. 287, pp. 195–203). https://doi.org/10.1007/978-3-319-07692-8_19

Mahdavi, M., Fesanghary, M., & Damangir, E. (2007a). An improved harmony search algorithm for solving optimization problems. *Applied Mathematics and*

Computation, *188*(2), 1567–1579. https://doi.org/10.1016/j.amc.2006.11.033

Mahdavi, M., Fesanghary, M., & Damangir, E. (2007b). An improved harmony search algorithm for solving optimization problems. *Applied Mathematics and Computation*, *188*(2), 1567–1579. https://doi.org/10.1016/j.amc.2006.11.033

Mallipeddi, R., Das, S., & Suganthan, P. N. (2015). *Ensemble of Constraint Handling Techniques for Single Objective Constrained Optimization* (pp. 231–248). https://doi.org/10.1007/978-81-322-2184-5_9

Marte, M. (2002). Models and Algorithms for School Timetabling – A Constraint-Programming Approach. In *Constraints*.

Martí, R., Sevaux, M., & Sörensen, K. (2024). 50 years of metaheuristics. *European Journal of Operational Research*. https://doi.org/10.1016/j.ejor.2024.04.004

McNamara, J. F. (1971). Mathematical Programming Models in Educational Planning. *Review of Educational Research*, *41*(5), 419–446. https://doi.org/10.3102/00346543041005419

Melício, F., Caldeira, J. P., & Rosa, A. (2006). THOR: A tool for school timetabling. *Practice and Theory of Automated Timetabling*, 532–535.

Minh, K. N. T. T., Thanh, N. D. T., Trang, K. T., & Hue, N. T. T. (2010). Using Tabu Search for Solving a High School Timetabling Problem. In *Studies in Computational Intelligence* (Vol. 283, pp. 305–313). https://doi.org/10.1007/978-3-642-12090-9_26

Misni, F., & Lee, L. S. (2021). Modified Harmony Search Algorithm for Location-Inventory-Routing Problem in Supply Chain Network Design with Product Returns. *Malaysian Journal of Mathematical Sciences*, *15*(1), 1–20.

Moerland, T. M., Broekens, J., Plaat, A., & Jonker, C. M. (2023). Model-based Reinforcement Learning: A Survey. *Foundations and Trends® in Machine Learning*, *16*(1), 1–118. https://doi.org/10.1561/2200000086

Molina, D., Poyatos, J., Ser, J. Del, García, S., Hussain, A., & Herrera, F. (2020). Comprehensive Taxonomies of Nature- and Bio-inspired Optimization: Inspiration Versus Algorithmic Behavior, Critical Analysis Recommendations. *Cognitive Computation*, *12*(5), 897–939. https://doi.org/10.1007/s12559-020-09730-8

Moral, M., Tabien, L. A., Napoles, F. M., Nunez, J. C., & Guevara, C. B. (2023). Effects of Class Schedules on the Focus and Punctuality of Senior High School Students. *Edukasiana: Jurnal Inovasi Pendidikan*, *2*(3), 204–221.

https://doi.org/10.56916/ejip.v2i3.403

Moreira, N. T. (2015). *A MIP-Based Approach to Solve a Real-World School Timetabling Problem* (Issue May 2015).

Moura, A. V., & Scaraficci, R. A. (2010). A GRASP strategy for a more constrained School Timetabling Problem. *International Journal of Operational Research*, *7*(2), 152. https://doi.org/10.1504/IJOR.2010.030801

Mushi, A. (2011). Non-Linear Great Deluge Algorithm for Tanzanian High Schools Timetabling. *International Journal of Advanced Research in Computer Science*, *2*(4).

Ngoc, N., Xuan, L., & Le, H. (2014). An integer programming formulation for a class of real-life school timetabling problems. *European Journal of Operational Research*.

Omran, M. G. H., & Mahdavi, M. (2008). Global-best harmony search. *Applied Mathematics and Computation*, *198*(2), 643–656. https://doi.org/10.1016/j.amc.2007.09.004

Ovrutsky, A. M., Prokhoda, A. S., & Rasshchupkyna, M. S. (2014). *Computational Materials Science*. Elsevier. https://doi.org/10.1016/C2013-0-13601-X

Oweiss, K. G. (2010). *Statistical Signal Processing for Neuroscience and Neurotechnology*. Elsevier. https://doi.org/10.1016/C2009-0-20215-7

Pandey, H. M. (2022). State of the art. In *State of the Art on Grammatical Inference Using Evolutionary Method* (pp. 35–124). Elsevier. https://doi.org/10.1016/B978-0-12-822116-7.00005-7

Papoutsis, K., Valouxis, C., & Housos, E. (2003). A column generation approach for the timetabling problem of Greek high schools. *Journal of the Operational Research Society*, *54*(3), 230–238. https://doi.org/10.1057/palgrave.jors.2601495

Patrick, J. (2012). A Markov decision model for determining optimal outpatient scheduling. *Health Care Management Science*, *15*(2), 91–102. https://doi.org/10.1007/s10729-011-9185-4

Peraza, C., Valdez, F., Garcia, M., Melin, P., & Castillo, O. (2016). A New Fuzzy Harmony Search Algorithm Using Fuzzy Logic for Dynamic Parameter Adaptation. *Algorithms*, *9*(4), 69. https://doi.org/10.3390/a9040069

Pillay, N. (2010). A study into the use of hyper-heuristics to solve the school timetabling problem. *Proceedings of the 2010 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists*

on - SAICSIT '10, 258–264. https://doi.org/10.1145/1899503.1899532

Pillay, N. (2014). A survey of school timetabling research. *Annals of Operations Research*, *218*(1), 261–293. https://doi.org/10.1007/s10479-013-1321-8

Pisinger, D., & Ropke, S. (2019). Large neighborhood search. In *International Series in Operations Research and Management Science* (Vol. 272, pp. 99–127). https://doi.org/10.1007/978-3-319-91086-4_4

Plaat, A. (2024). Research re: Search & re-search. *ArXiv Preprint ArXiv:2403.13705*.

Post, G., Ahmadi, S., Daskalaki, S., Kingston, J. H., Kyngas, J., Nurmi, C., & Ranson, D. (2012). An XML format for benchmarks in High School Timetabling. *Annals of Operations Research*, *194*(1), 385–397. https://doi.org/10.1007/s10479-010-0699-9

Post, G., Kingston, J. H., Ahmadi, S., Daskalaki, S., Gogos, C., Kyngas, J., Nurmi, C., Musliu, N., Pillay, N., Santos, H., & Schaerf, A. (2014). XHSTT: An XML archive for high school timetabling problems in different countries. *Annals of Operations Research*, *218*(1), 295–301. https://doi.org/10.1007/s10479-011-1012-2

Qu, R., Burke, E. K., McCollum, B., Merlot, L. T. G., & Lee, S. Y. (2009). A survey of search methodologies and automated system development for examination timetabling. *Journal of Scheduling*, *12*(1), 55–89. https://doi.org/10.1007/s10951-008-0077-5

Quan, Y., Ouyang, H., Zhang, C., Li, S., & Gao, L.-Q. (2021). Mobile Robot Dynamic Path Planning Based on Self-Adaptive Harmony Search Algorithm and Morphin Algorithm. *IEEE Access*, *9*, 102758–102769. https://doi.org/10.1109/ACCESS.2021.3098706

Raghavjee, R., & Pillay, N. (2008). An application of genetic algorithms to the school timetabling problem. *ACM International Conference Proceeding Series*, *338*(November), 193–199. https://doi.org/10.1145/1456659.1456682

Raghavjee, R., & Pillay, N. (2010). An informed genetic algorithm for the high school timetabling problem. *Proceedings of the 2010 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on - SAICSIT '10*, 408–412. https://doi.org/10.1145/1899503.1899555

Raghavjee, R., & Pillay, N. (2012). A comparison of genetic algorithms and genetic programming in solving the school timetabling problem. *2012 Fourth World Congress on Nature and Biologically Inspired Computing (NaBIC)*, 98–103.

https://doi.org/10.1109/NaBIC.2012.6402246

Raghavjee, R., & Pillay, N. (2013). A Study of Genetic Algorithms to Solve the School Timetabling Problem. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Vol. 8266 LNAI* (Issue PART 2, pp. 64–80). https://doi.org/10.1007/978-3-642-45111-9_6

Rahimi, I., Gandomi, A. H., Chen, F., & Mezura-Montes, E. (2023). A Review on Constraint Handling Techniques for Population-based Algorithms: from single-objective to multi-objective optimization. *Archives of Computational Methods in Engineering*, *30*(3), 2181–2209. https://doi.org/10.1007/s11831-022-09859-9

Reddy, G. T., Reddy, M. P. K., Lakshmanna, K., Rajput, D. S., Kaluri, R., & Srivastava, G. (2020). Hybrid genetic algorithm and a fuzzy logic classifier for heart disease diagnosis. *Evolutionary Intelligence*, *13*(2), 185–196. https://doi.org/10.1007/s12065-019-00327-1

Reddy Madhavi, K., Vinay Kumar Reddy, B., Vinay, D., Sai Pranav, D. V., & Patan, R. (2024). *Adopting Harmony Search Algorithm in Deep Learning* (pp. 611–616). https://doi.org/10.1007/978-981-99-2832-3_71

Ren, H., Chen, X., & Chen, Y. (2017). Chapter 11 - The Methodologies of Reliability and Maintainability in the A380 Program. In H. Ren, X. Chen, & Y. B. T.-R. B. A. M. O. and A. Chen (Eds.), *Aerospace Engineering* (pp. 207–222). Academic Press. https://doi.org/https://doi.org/10.1016/B978-0-12-812668-4.00011-3

Rezaee Jordehi, A., & Jasni, J. (2015). Particle swarm optimisation for discrete optimisation problems: a review. *Artificial Intelligence Review*, *43*(2), 243–258. https://doi.org/10.1007/s10462-012-9373-8

Riquelme-Dominguez, J. M., Acosta, M. N., Gonzalez-Longatt, F., Andrade, M. A., Vazquez, E., & Rueda, J. L. (2022). Improved Harmony Search Algorithm to Compute the Underfrequency Load Shedding Parameters. *International Transactions on Electrical Energy Systems*, *2022*, 1–15. https://doi.org/10.1155/2022/5381457

Rizk, Y., Awad, M., & Tunstel, E. W. (2018). Decision Making in Multiagent Systems: A Survey. *IEEE Transactions on Cognitive and Developmental Systems*, *10*(3), 514–529. https://doi.org/10.1109/TCDS.2018.2840971

Roth, L., & Corsi, S. (2023). Ambidexterity in a geographic context: A systematic literature review on international exploration and exploitation of knowledge.

Technovation, *124*, 102744. https://doi.org/10.1016/j.technovation.2023.102744

Rousseeuw, P. J., & Leroy, A. M. (2005). *Robust regression and outlier detection*. John wiley & sons.

Russell, S. J., Norvig, P., & Davis, E. (2010). *Artificial Intelligence: A Modern Approach*. Prentice Hall. https://books.google.co.th/books?id=8jZBksh-bUMC

Şahin, İ. (2017). Markov chain model for delay distribution in train schedules: Assessing the effectiveness of time allowances. *Journal of Rail Transport Planning & Management*, *7*(3), 101–113. https://doi.org/10.1016/j.jrtpm.2017.08.006

Sanjay R, S., & Rajan S, B. (2017). HIGH SCHOOL TIMETABLING USING TABU SEARCH AND PARTIAL FEASIBILITY PRESERVING GENETIC ALGORITHM. *International Journal of Advances in Engineering & Technology*, *10*(3), 421–428.

Santos, H. G., Ochi, L. S., & Souza, M. J. F. (2005). A Tabu search heuristic with efficient diversification strategies for the class/teacher timetabling problem. *ACM Journal of Experimental Algorithmics*, *10*(2), 1–16. https://doi.org/10.1145/1064546.1180621

Santos, H. G., Uchoa, E., Ochi, L. S., & Maculan, N. (2012). Strong bounds with cut and column generation for class-teacher timetabling. *Annals of Operations Research*, *194*(1), 399–412. https://doi.org/10.1007/s10479-010-0709-y

Saraswati, N. W. S., Artakusuma, I. D. M. D., & Indradewi, I. G. A. A. D. (2021). Modified genetic algorithm for employee work shifts scheduling optimization. *Journal of Physics: Conference Series*, *1810*(1), 012014. https://doi.org/10.1088/1742-6596/1810/1/012014

Saviniec, L., & Constantino, A. A. (2017). Effective local search algorithms for high school timetabling problems. *Applied Soft Computing*, *60*, 363–373. https://doi.org/10.1016/j.asoc.2017.06.047

Saviniec, L., Constantino, A. A., & Romão, W. (2013). Vns based algorithms to the high school timetabling problem. *Anais Do XLV Simpósio Brasileiro de Pesquisa Operacional*, 845–856.

Saviniec, L., Santos, M. O., & Costa, A. M. (2018). Parallel local search algorithms for high school timetabling problems. *European Journal of Operational Research*, *265*(1), 81–98. https://doi.org/10.1016/j.ejor.2017.07.029

Saviniec, L., Santos, M. O., Costa, A. M., & Santos, L. M. R. dos. (2020). Pattern-

based models and a cooperative parallel metaheuristic for high school timetabling problems. *European Journal of Operational Research*, *280*(3), 1064–1081. https://doi.org/10.1016/j.ejor.2019.08.001

Schaerf, A. (1999). Survey of automated timetabling. *Artificial Intelligence Review*, *13*(2), 87–127. https://doi.org/https://doi.org/10.1023/A:1006576209967

Schiewe, P. (2020). *Integrating Timetabling and Vehicle Scheduling* (pp. 91–98). https://doi.org/10.1007/978-3-030-46270-3_4

Schneider, P., Afzal, A., Vladika, J., Braun, D., & Matthes, F. (2023). *Investigating Conversational Search Behavior for Domain Exploration* (pp. 608–616). https://doi.org/10.1007/978-3-031-28238-6_52

Seabrook, E., & Wiskott, L. (2022). *A Tutorial on the Spectral Theory of Markov Chains*. 0–2. https://doi.org/10.48550/ARXIV.2207.02296

Shambour, M. K. Y., Khader, A. T., Kheiri, A., & Özcan, E. (2013). A Two Stage Approach for High School Timetabling. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Vol. 8226 LNCS* (Issue PART 1, pp. 66–73). https://doi.org/10.1007/978-3-642-42054-2_9

Silva, J., Cabrera, D., Maco, J., Villón, M., Guliany, J. G., Roncallo, A., & Palma, H. H. (2020). Comparison of Bioinspired Algorithms Applied to the Timetabling Problem in Sport. *Procedia Computer Science*, *170*. https://doi.org/10.1016/j.procs.2020.03.100

Silveira, C. S., Oliveira, M. O. R. de, Heldt, R., & Luce, F. B. (2020). Trade-off between value creation and value appropriation? *Marketing Intelligence & Planning*, *39*(1), 1–16. https://doi.org/10.1108/MIP-11-2019-0592

Skoullis, V. I., Tassopoulos, I. X., & Beligiannis, G. N. (2017). Solving the high school timetabling problem using a hybrid cat swarm optimization based algorithm. *Applied Soft Computing*, *52*, 277–289. https://doi.org/10.1016/j.asoc.2016.10.038

Smith, K. A., Abramson, D., & Duke, D. (2003). Hopfield neural networks for timetabling: Formulations, methods, and comparative results. *Computers and Industrial Engineering*, *44*(2), 283–305. https://doi.org/10.1016/S0360-8352(02)00180-8

Somashekar, T., & Jagirdar, S. (2024). A Hybrid Feature Selection Framework Using Opposition-Based Harmony Search and Manta Ray Foraging Optimization.

Journal of Advances in Information Technology, *8*, 982–990. https://doi.org/10.12720/jait.15.8.982-990

Sørensen, M. (2013). *Timetabling at High Schools*.

Sørensen, M., & Dahms, F. H. W. (2014). A Two-Stage Decomposition of High School Timetabling applied to cases in Denmark. *Computers & Operations Research*, *43*(1), 36–49. https://doi.org/10.1016/j.cor.2013.08.025

Sørensen, M., Kristiansen, S., & Stidsen, T. R. (2012). International timetabling competition 2011: An Adaptive Large Neighborhood Search algorithm. *PATAT 2012 - Proceedings of the 9th International Conference on the Practice and Theory of Automated Timetabling*, 489–492.

Sorensen, M., & Stidsen, T. J. R. (2013). Comparing solutions approches for a complete model of high school timetabling. *DTU Management Engineering Report*, *5*(2013), 1–45.

Sørensen, M., & Stidsen, T. R. (2012). High school timetabling: Modeling and solving a large number of cases in Denmark. *PATAT 2012 - Proceedings of the 9th International Conference on the Practice and Theory of Automated Timetabling*, 365–368.

Su, F., Wu, J., & He, S. (2019). Set pair analysis-Markov chain model for groundwater quality assessment and prediction: A case study of Xi'an city, China. *Human and Ecological Risk Assessment: An International Journal*, *25*(1–2), 158–175. https://doi.org/10.1080/10807039.2019.1568860

Sutar, S. R., & Bichkar, R. S. (2016). Genetic Algorithms based Timetabling using Knowledge Augmented Operators. *International Journal of Computer Science and Information Security*, *14*(11), 570.

Szwarc, K., & Boryczka, U. (2022). A novel approach to the Orienteering Problem based on the Harmony Search algorithm. *PLOS ONE*, *17*(2), e0264584. https://doi.org/10.1371/journal.pone.0264584

Tan, J. S., Goh, S. L., Kendall, G., & Sabar, N. R. (2021). A survey of the state-of-the-art of optimisation methodologies in school timetabling problems. *Expert Systems with Applications*, *165*(August 2020), 113943. https://doi.org/10.1016/j.eswa.2020.113943

Tan, J. S., Goh, S. L., Sura, S., Kendall, G., & Sabar, N. R. (2020). Hybrid particle swarm optimization with particle elimination for the high school timetabling problem. *Evolutionary Intelligence*, *0123456789*.

https://doi.org/10.1007/s12065-020-00473-x

Tan, J. S., Goh, S. L., Sura, S., Kendall, G., & Sabar, N. R. (2021). Hybrid particle swarm optimization with particle elimination for the high school timetabling problem. *Evolutionary Intelligence*, *14*(4), 1915–1930. https://doi.org/10.1007/s12065-020-00473-x

Tao, L., Ma, J., Cheng, Y., Noktehdan, A., Chong, J., & Lu, C. (2017). A review of stochastic battery models and health management. *Renewable and Sustainable Energy Reviews*, *80*, 716–732. https://doi.org/10.1016/j.rser.2017.05.127

Tassopoulos, I. X., & Beligiannis, G. N. (2012a). Solving effectively the school timetabling problem using particle swarm optimization. *Expert Systems with Applications*, *39*(5), 6029–6040. https://doi.org/10.1016/j.eswa.2011.12.013

Tassopoulos, I. X., & Beligiannis, G. N. (2012b). Solving effectively the school timetabling problem using particle swarm optimization. *Expert Systems with Applications*, *39*(5), 6029–6040. https://doi.org/10.1016/j.eswa.2011.12.013

Tassopoulos, I. X., Iliopoulou, C. A., & Beligiannis, G. N. (2020). Solving the Greek school timetabling problem by a mixed integer programming model. *Journal of the Operational Research Society*, *71*(1), 117–132. https://doi.org/10.1080/01605682.2018.1557022

Teixeira, U. R., Souza, M. J. F., de Souza, S. R., & Coelho, V. N. (2019). An Adaptive VNS and Skewed GVNS Approaches for School Timetabling Problems. In M. Gendreau & J.-Y. Potvin (Eds.), *Handbook of Metaheuristics* (Vol. 146, pp. 101–113). Springer US. https://doi.org/10.1007/978-3-030-15843-9_9

Tsirtsis, S., Tabibian, B., Khajehnejad, M., Singla, A., Schölkopf, B., & Gomez-Rodriguez, M. (2024). Optimal Decision Making Under Strategic Behavior. *Management Science*. https://doi.org/10.1287/mnsc.2021.02567

Uray, E., Carbas, S., Geem, Z. W., & Kim, S. (2022). Parameters Optimization of Taguchi Method Integrated Hybrid Harmony Search Algorithm for Engineering Design Problems. *Mathematics*, *10*(3), 327. https://doi.org/10.3390/math10030327

Valouxis, C., Gogos, C., Alefragis, P., & Housos, E. (2012). Decomposing the High School Problem. *The 9th International Conference on the Practice and Theory of Automated Timetabling, Son Norway*, *August*, 29–31.

van Stein, N., & Bäck, T. (2024). *LLaMEA: A Large Language Model Evolutionary*

Algorithm for Automatically Generating Metaheuristics. http://arxiv.org/abs/2405.20132

Veenstra, M., & Vis, I. F. A. (2016). School timetabling problem under disturbances. *Computers & Industrial Engineering*, *95*(February), 175–186. https://doi.org/10.1016/j.cie.2016.02.011

Vitor, F. T. (2019). *Two dimensional search algorithms for linear programming*.

Wahid, J. (2017). *Hybridizing harmony search with local search based metaheuristic for solving curriculum based university course timetabling*. Institute of Graduate Studies, UiTM.

Wahid, J., Abdul-Rahman, S., Din, A. M., Yusof, S. M., & Hussain, N. M. (2018). Solving Curriculum Based University Course Timetabling Problem: Trends on Metaheuristic Approaches. *International Conference on ICT for Transformation (IC-ICT4T2018)*, *3*(5), 42–47.

Wahid, J., Abdul-Rahman, S., Mohamed Din, A., & Mohd-Hussin, N. (2019). Constructing population of initial university timetable: Design and analysis. *Indonesian Journal of Electrical Engineering and Computer Science*, *15*(2), 1109–1118. https://doi.org/10.11591/ijeecs.v15.i2.pp1109-1118

Wahid, J., & Hussin, N. M. (2014). Harmony Search Algorithm for Curriculum-Based Course Timetabling Problem. *International Journal of Soft Computing and Software Engineering [JSCSE]*, *3*(3), 365–371. https://doi.org/10.7321/jscse.v3.n3.55

Wahid, J., & Hussin, N. M. (2017). Hybrid harmony search with great deluge for UUM CAS curriculum based course timetabling. *Journal of Telecommunication, Electronic and Computer Engineering*, *9*(1–2), 33–38.

Wang, C.-M., & Huang, Y.-F. (2010). Self-adaptive harmony search algorithm for optimization. *Expert Systems with Applications*, *37*(4), 2826–2837. https://doi.org/10.1016/j.eswa.2009.09.008

Wang, F., Zhang, H., & Zhou, A. (2021). A particle swarm optimization algorithm for mixed-variable optimization problems. *Swarm and Evolutionary Computation*, *60*, 100808. https://doi.org/10.1016/j.swevo.2020.100808

Wang, J., Ouyang, H., Zhang, C., Li, S., & Xiang, J. (2023). A novel intelligent global harmony search algorithm based on improved search stability strategy. *Scientific Reports*, *13*(1), 7705. https://doi.org/10.1038/s41598-023-34736-1

Wang, J., Ouyang, H., Zhou, Z., & Li, S. (2023). Harmony Search Algorithm Based

on Dual-Memory Dynamic Search and Its Application on Data Clustering. *Complex System Modeling and Simulation*, *3*(4), 261–281. https://doi.org/10.23919/CSMS.2023.0019

Wang, L., Hu, H., Liu, R., & Zhou, X. (2019). An improved differential harmony search algorithm for function optimization problems. *Soft Computing*, *23*(13), 4827–4852. https://doi.org/10.1007/s00500-018-3139-4

Wilke, P., Gröbner, M., & Oster, N. (2002). A Hybrid Genetic Algorithm for school timetabling. *Lecture Notes in Artificial Intelligence (Subseries of Lecture Notes in Computer Science)*, *2557*, 455–464. https://doi.org/10.1007/3-540-36187-1_40

Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, *1*(1), 67–82. https://doi.org/10.1109/4235.585893

Wren, A. (1996). Scheduling, timetabling and rostering — A special relationship? In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (Vol. 1153, pp. 46–75). https://doi.org/10.1007/3-540-61794-9_51

Xu, J., Yan, F., Yun, K., Ronald, S., Li, F., & Guan, J. (2019). Dynamically Dimensioned Search Embedded with Piecewise Opposition-Based Learning for Global Optimization. *Scientific Programming*, *2019*, 1–20. https://doi.org/10.1155/2019/2401818

Yang, X.-S. (2020). *Nature-Inspired Computation and Swarm Intelligence*. Elsevier. https://doi.org/10.1016/C2019-0-00628-0

Yang, X.-S. (2021). *Nature-Inspired Optimization Algorithms*. Elsevier. https://doi.org/10.1016/C2019-0-03762-4

Yang, X.-S. (2024). *A Generalized Evolutionary Metaheuristic (GEM) Algorithm for Engineering Optimization*. https://doi.org/10.1080/23311916.2024.2364041

Yang, X.-S., He, X.-S., & Fan, Q.-W. (2020). Mathematical framework for algorithm analysis. In *Nature-Inspired Computation and Swarm Intelligence* (pp. 89–108). Elsevier. https://doi.org/10.1016/B978-0-12-819714-1.00017-8

Ye, Y., Grossmann, I. E., Pinto, J. M., & Ramaswamy, S. (2019). Modeling for reliability optimization of system design and maintenance based on Markov chain theory. *Computers & Chemical Engineering*, *124*, 381–404. https://doi.org/10.1016/j.compchemeng.2019.02.016

Yildiz, A. R., & Öztürk, F. (2010). *Hybrid Taguchi-Harmony Search Approach for*

Shape Optimization (pp. 89–98). https://doi.org/10.1007/978-3-642-04317-8_8

Yoshikawa, M., Kaneko, K., Nomura, Y., & Watanabe, M. (1994). Constraint-based approach to high-school timetabling problems: A case study. *Proceedings of the National Conference on Artificial Intelligence*, *2*, 1111–1116. https://doi.org/https://dl.acm.org/doi/10.5555/2891730.2891902

Yu, X., & Gen, M. (2010). *Introduction to Evolutionary Algorithms*. Springer London. https://doi.org/10.1007/978-1-84996-129-5

Zhang, D., Liu, Y., M'Hallah, R., & Leung, S. C. H. (2010). A simulated annealing with a new neighborhood structure based algorithm for high school timetabling problems. *European Journal of Operational Research*, *203*(3), 550–558. https://doi.org/10.1016/j.ejor.2009.09.014

Zhang, W., Du, W., Yu, G., He, R., Du, W., & Jin, Y. (2024). *Knowledge-Assisted Dual-Stage Evolutionary Optimization of Large-Scale Crude Oil Scheduling*. http://arxiv.org/abs/2401.10274

Zong Woo Geem, Joong Hoon Kim, & Loganathan, G. V. (2001). A New Heuristic Optimization Algorithm: Harmony Search. *SIMULATION*, *76*(2), 60–68. https://doi.org/10.1177/003754970107600201