ADAPTING AND HYBRIDISING HARMONY SEARCH WITH METAHEURISTIC COMPONENTS FOR UNIVERSITY COURSE TIMETABLING

MOHAMMED AZMI AL-BETAR

UNIVERSITI SAINS MALAYSIA

2010

ADAPTING AND HYBRIDISING HARMONY SEARCH WITH METAHEURISTIC COMPONENTS FOR UNIVERSITY COURSE TIMETABLING

by

MOHAMMED AZMI AL-BETAR

Thesis submitted in fulfilment of the requirements
for the degree of
Doctor of Philosophy

June 2010

ACKNOWLEDGEMENTS

I am very grateful and thankful to Allah S.W.T for giving me strength to complete my research study. Although I am solely responsible for the study and its findings, I must acknowledge many external contributions coming from people who extended a helping hand throughout this research.

I am much indebted to my academic supervisor, Associated Professor Dr. Ahamad Tajudin Khader from the School of Computer Sciences at Universiti Sains Malaysia, for his wise counseling, valuable advice, continuous support, help and guidance throughout the duration of my research. Indeed, without his support and cooperation, I could not have completed this study. I owe my deepest gratitude to my co-supervisor, Dr. Munir Zaman from the School of Computer Sciences, Universiti Sains Malaysia, for his support in a number of ways especially in the technical aspects and presentation of this dissertation. I would also like to thank Dr. Iman Yi Liao for her reviewing of some mathematical notations. I am grateful to Universiti Sains Malaysia for financial support under the Fellowship Scheme throughout my Ph.D candidature.

I am also obliged to Mr. Osama Bitar, school supervisor of English at UNRWA, Jordan for his valuable comments, suggestions, and helping out with the English proof reading. My thanks go to my beloved parents for their love, patience, encouragement, and continuous support. Words alone cannot express the thanks I owe to my beloved wife Huda Abdul-Rahman Habboush, Master of English, who extended a helping hand in writing my dissertation, and supported me heart and soul all through this tiring research task while at the same time taking care of our children Ahamad and Amneh.

Last but not least, I thank those who supported me in any respect during my research, especially my friends Ali Kattan, Osama Alia, Khaid Jaber, Mohammed Abul-Rub and many others that whose names I can not recall.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ACO Ant Colony Optimisation

AIS Artificial Immune System

ANN Artificial Neural Network

bw bandwidth parameter

COP Combinatorial Optimisation Problem

EC Evolutionary Computation

EDA Estimation of Distribution Algorithm

EP Evolutionary Programming

ES Evolutionary Strategies

GA Genetic Algorithm

GD Great Deluge

GLS Guided Local Search

GP Genetic Programming

GRASP Greedy Randomized Adaptive Search Procedure

HCO Hill Climbing Optimiser

HCR Hill Climbing Rate

HHSA Hybrid Harmony Search Algorithm

HM Harmony Memory

HMCR Harmony Memory Consideration Rate

HMS Harmony Memory Size

HSA Harmony Search Algorithm

HSA-MPAR Harmony Search Algorithm with Multi-Pitch Adjusting Rate

ILS Iterated Local Search

ITC-2007 Second International Timetabling Competition

MA Memetic Algorithm

MHSA Modified Harmony Search Algorithm

MMAS MAX-MIN Ant System

NP Non Polynomial

PAR Pitch adjusting Rate

PSO Particle Swarm Optimisation

PSR Particle Swarm Rate

PV Penalty Value

SA Simulated Annealing

Scen Scenario

SI Swarm Intelligence

SS Scatter Search

TS Tabu Search

TTComp2002 First International Timetabling Competition

UCTP University Course Timetabling Problem

USM Universiti Sains Malaysia

UTP University Timetabling problem

VNS Variable Neighborhood Search

LIST OF PUBLICATIONS

1. Al-Betar, M.A., Khader, A.T.: A Harmony Search Algorithm for University Course Timetabling. Annals of Operation Research, DOI: 10.1007/s10479-010-0769-z.

2. Al-Betar, M.A., Khader, A.T.: A MultiSwap Algorithm for University Course Timetabling. Journal of Operation Research Society, **Under revision**.

3. Al-Betar, M.A., Khader A.T., and Liao I.Y.: A Harmony Search Algorithm with Multi-Pitch Adjusting Rate for University Course Timetabling. In Z.W. Geem, editor, Recent Advances in Harmony Search Algorithm, volume 270 of Studies in Computational Intelligence (SCI), pages 147–162. Springer-Verlag, Berlin, Heidelberg (2010).

4. Al-Betar, M.A., Khader, A.T., Gani, T.A.: A Harmony Search Algorithm for University Course Timetabling. In: 7th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2008), Montreal, Canada, August 18–22 (2008).

5. Gani, T.A., Khader, A.T., Al-Betar, M.A.: Assessing examination timetabling problems using fuzzy Pareto optimality. In: 7th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2008), Montreal, Canada, August 18–22 (2008).

6. Al-Betar, M.A., Khader A.T., Nadi, F.: Selection Mechanisms in Memory Consideration for Examination Timetabling with Harmony Search. In Proceedings of Genetic and Evolutionary Computation Conference (GECCO-2010). Portland, Oregon, USA, 7-11 July (2010).

7. Nadi, F., Khader A.T., Al-Betar, M.A: Adaptive Genetic Algorithm Using Harmony Search. In Proceedings of Genetic and Evolutionary Computation Conference (GECCO-2010). Portland, Oregon, USA, 7-11 July (2010).

8. Al-Betar, M.A., Khader, A.T., Thomas, J. J.: A Combination of Metaheuristic Components based on Harmony Search for the Uncapacitated Examination Timetabling. In: 8th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2010), Belfast, Northern Ireland, August 10–13 (2010).

9. Thomas, J. J., Khader, A.T., Al-Betar, M.A. : The Perception of Interaction on the University Examination Timetabling Problem. In: 8th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2010), Belfast, Northern Ireland, August 10–13 (2010).

10. Al-Betar, M.A., Khader A.T.: A hybrid harmony search algorithm for university course timetabling. In Proceedings 4nd Multidisciplinary Conference on Scheduling: Theory and Applications (MISTA2009). Dublin, Ireland, 10-12 August (2009).

PENYESUAIAN DAN PENGHIBRIDAN GELINTARAN HARMONI DENGAN KOMPONEN METAHEURISTIK UNTUK PENJADUALAN KURSUS UNIVERSITI

ABSTRAK

Masalah Penjadualan Waktu Kursus Universiti (MPWKU) merupakan suatu masalah penjadualan kombinatorik yang rumit. Algoritma Gelintaran Harmoni (AGH) ialah suatu kaedah metaheuristik berdasarkan populasi. Kelebihan utama algoritma ini terletak pada keupayaannya dalam mengintegrasikan komponen-komponen utama bagi kaedah berdasarkan populasi dan kaedah berdasarkan gelintaran setempat dalam satu model pengoptimuman yang sama. Disertasi ini mencadangkan suatu AGH yang telah disesuaikan untuk MPWKU. Penyesuaian ini melibatkan pengubahsuaian terhadap operator AGH. Hasil yang diperoleh adalah dalam lingkungan keputusan terdahulu. Tetapi beberapa kelemahan dalam kadar penumpuan dan eksploitasi setempat telah dikesan dan telah diberikan tumpuan menerusi penghibridan dengan komponen metaheuristik yang diketahui. Tiga versi terhibrid dicadangkan, di mana, setiap hibrid merupakan peningkatan daripada yang sebelumnya: (i) Algoritma Gelintaran Harmoni yang Diubah suai; (ii) Algoritma Gelintaran Harmoni dengan Kadar Penyesuaian Berbagai Nada, dan (iii) Algoritma Gelintaran Harmoni Hibrid. Semua hasil yang diperoleh dibandingkan dengan 21 kaedah lain menggunakan sebelas dataset piawai *de facto* yang mempunyai saiz dan kekompleksan yang berbeza-beza. Versi terhibrid yang dicadangkan ini berjaya memberikan penyelesaian optimal bagi dataset kecil, dengan dua hasil keseluruhan terbaik bagi dataset sederhana. Seterusnya, dalam dataset yang besar dan paling kompleks kaedah hibrid yang dicadangkan ini telah menghasilkan keputusan terbaik.

ADAPTING AND HYBRIDISING HARMONY SEARCH WITH METAHEURISTIC COMPONENTS FOR UNIVERSITY COURSE TIMETABLING

ABSTRACT

University Course Timetabling Problem (UCTP) is a hard combinatorial scheduling problem. Harmony Search Algorithm (HSA) is a recent metaheuristic population-based method. The major thrust of this algorithm lies in its ability to integrate the key components of population-based methods and local search-based methods in the same optimisation model. This dissertation presents a HSA adapted for UCTP. The adaptation involved modifying the HSA operators. The results were within the range of state of the art. However, some shortcomings in the convergence rate and local exploitation were identified and addressed through hybridisation with known metaheuristic components. Three hybridized versions are proposed which are incremental improvements over the preceding version: (i) Modified Harmony Search Algorithm (MHSA); (ii) Harmony Search Algorithm with Multi-Pitch Adjusting Rate (HSA-MPAR), and (iii) Hybrid Harmony Search Algorithm (HHSA). The results were compared against 21 other methods using eleven *de facto* standard dataset of different sizes and complexity. The proposed hybridized versions achieved the optimal solution for the small datasets, with two best overall results for the medium datasets. Furthermore, in the large and most complex dataset the proposed hybrid methods achieved the best result.

CHAPTER 1

INTRODUCTION

1.1 Background

1.1.1 Timetabling

Timetabling is a process observed by administrators in different disciplines for the purpose of assigning events to be carried out at an appointed time. The overall aim is to organise people's lives, work and activities that are desired to be performed timely and without disruption. Transportation sectors, hospitals, companies, and academic institutions are but a few examples of the bodies that have to run business within a framework of timetabling to ensure smooth transactions.

For those working in the administrative divisions and the registration departments at universities and academic institutions, the timetabling task may seem monumental, overtiring, and even baffling. So much so that some of them have to spend nights and weeks in an attempt to overcome certain intricacies arising from the need to deal with huge data to be organized neatly and systematically. This is so because they have to ensure that students attend their classes smoothly without conflict or confusion taking into consideration student and/or lecturer preferences.

The university administrators who are in charge of the process of drawing up a timetable need to assess what resources they have and the requirements that they need to satisfy. The resources may include thousands of students who attend a limited number of courses, a limited number of time periods, and a limited number of lecture halls (or rooms). The main activity is

to assign these courses to time periods and rooms according to predefined requirements. Some of these requirements have to be satisfied. For example, you cannot schedule two courses in the same period if they have the same students (i.e., a hard constraint). In addition, there are preferences which are not mandatory but rather desired to be satisfied. For example, students may not like to have three courses scheduled in a row (i.e., a soft constraint). In the university context, two types of timetables are generated in each semester: exam timetable and course timetable. This dissertation addresses course timetabling.

In technical terms, course timetabling is the process of allocating given events, each with given features, to given resources and times with respect to given constraints (Burke et al., 2004). The timetabling process varies in difficulty according to the problem size and demanding constraints which vary among academic institutions. The timetabling solution is typically evaluated against satisfying constraints which are usually categorized into two types: hard and soft. Hard constraints *must* be satisfied for the timetabling solution to be *feasible*, whereas soft constraints are desired but not absolutely essential. Soft constraints may be violated, yet the more they are met, the better the quality of the solution will be.

To clarify further, Figure 1.1 shows a real-world course timetable for the School of Computer Sciences at the University Sains Malaysia for the second semester of the academic year 2009/2010. Each *row* represents a day, each *column* represents a *time period* and each cell contains a value representing a scheduled course. The value in a cell consists of a course code with its required lecture hall size (e.g., Course CPT103 with 70 students for the cell "(CPT103)(70)") and the lecture hall name with its capacity (e.g., "DK R (110)", represents lecture room DK R with capacity of 110 students).

If the timetable is feasible, each course is allocated in a suitable room in terms of capacity and features (overhead projector, video Conferencing, etc.). Furthermore, the courses taken by

	8:00 am 8:50 am	9:00 am 9:50 am	10:00 am 10:50 am	11:00 am 11:50 am	12:00 pm 12:50 pm	1:00 pm 1:50 pm	2:00 pm 2:50 pm	3:00 pm 3:50 pm	4:00 pm 4:50 pm
Isnin		CMT325 (120) DK C (120)	CMM111 (700) DK G31 (300)	(CPT201) (300) DK G31 (300)	(CPT201) (300) DK G31 (300)		CPT103 (110) DK P (110)	CPT314 (70) DK R (110)	CMT315 (110) DK Q (110)
		CPT314 (70) BT 148 (70)	CPT104 (350) DK W (400)				CPT341 (70) DK N (110)		CPT317 (110) DK J (110)
			CPT341 (70) BT 141 (70)						CST315 (110) DK R (110)
Selasa	CPT315 (110) DK N (110)		(CMT202) (200) DK G31 (300)	(CPT104) (350) DK G31 (300)	(CPT313) (110) DK G31 (300)		CPT103 (110) DK M (110)	CMT325 (120) DK B (120)	CMT315 (110) DK Q (110)
	CST313 (110) DK K (110)		CMT316 (110) DK H (260)	CPT345 (70) BT 144 (70)	CPT115 (40) BT 153 (40)		CPT341 (70) BT 150 (70)	CPT314 (70) BT 150 (70)	CPT317 (110) DK N (110)
					CST311 (110) DK L (110)			CST231 (200) DK H (260)	
Rabu	(CMT202) (200) DK G31 (300)		CPT314 (70) DK G31 (300)	CMT312 (200) DK H (260)	CMT315 (110) DK V (200)		(CPT201) (300) DK G31 (300)		
		CMT316 (110) DK R (110)	CMT325 (120) DK C (120)	CST314 (110) DK SK3 (700)	CPT317 (110) DK I (110)		CMT314 (200) DK H (260)		
			CST231 (200) DK T (200)		CST315 (110) DK N (110)				
Khamis	CPT315 (110) DK Q (110)		CST335 (50) BT 104 (50)	(CPT104) (350) DK G31 (300)	(CPT115) (40) DK G31 (300)		CST314 (110) DK I (110)		
				CPT345 (70) BT 150 (70)	CPT313 (110) DK M (110)				
					CST311 (110) DK P (110)				
Jumaat	(CPT115) (40) DK G31 (300)							(CMT322) (110) DK G31 (300)	(CMT202) (200) DK G31 (300)
	CST132 (250) DK H (260)								CMT316 (110) DK I (110)

Figure 1.1: An Example of a *Feasible* Course Timetable

the same students are allocated in different day-periods. For example, on Monday (Isnin), for the time period 9:00 to 9:50 (Period 2); two courses are scheduled (i.e., CMT325 and CPT314). Therefore, no students can attend both of these courses and the same lecturer cannot teach both either. However, if a student were enrolled in CMT325 and CPT314, then the timetable would not be feasible, as a hard constraint would be violated. The lecturer for CMT325 may have difficulties coming on Monday morning (e.g., this lecturer may live faraway off-campus) and may not prefer to be scheduled in Period 2 – this is an example on a soft constraint which was not met. However, there is a possibility for another feasible timetable to be generated where the above preference might be satisfied (with no other constraints violated) – this will be evaluated as a higher quality solution.

1.1.2 Datasets

Evaluating a method for solving the course timetabling problem is conducted by instantiation with a dataset. A number of such datasets are publicly available including those provided

by the First International Timetabling Competition (TTComp2002)[1], the Second International Timetabling Competition (ITC-2007) (McCollum et al., 2010); and researchers Lewis and Paechter (2005) and Socha et al. (2002). These datasets provide a common framework for meaningful comparative evaluations of their methods. The datasets vary in terms of size (e.g., number of rooms, courses, room types) and constraints. For more details about these datasets, please refer to Appendix C.

This dissertation uses a *de facto* dataset defined by Socha et al. (2002). The Socha dataset has been used for evaluating a number of optimisation methods over the last several years. Although other datasets exist, the Socha dataset provides a suitable and wide range of comparative methods, numbering 21, against which the proposed methods have been comprehensively evaluated.

1.1.3 Course Timetabling Methods

The course timetabling problem has been given particular attention by operational research and artificial intelligence researchers for some time (Burke and Petrovic, 2002). Many methods have been introduced in the literature to tackle such problems. Interestingly, the basic timetabling problem can be modeled as a graph coloring problem. Therefore, the earliest methods employed *graph coloring heuristics* as an essential part to construct the course timetabling solution (Burke et al., 2004). These heuristics assign courses to rooms and timeslots, individually, according to a particular order. Although these heuristics show great efficiency in constructing a timetabling solution quickly, the quality of the solution is often inferior to that produced by metaheuristic or hyper-heuristic methods. Nowadays, graph coloring heuristics tend to be used in the construction of initial solution for metaheuristic methods (Abdullah et al., 2007b; Chiarandini et al., 2006).

[1] http://www.idsia.ch/Files/ttcomp2002/

The emergence of *metaheuristics* for solving difficult timetabling problems has been one of the most notable accomplishments over almost the last two decades (Lewis, 2008). Metaheuristic-based method is an iterative improvement process that uses its operators and combines intelligently the problem specific knowledge for exploring and exploiting the search space in order to reach a good-quality solution (Osman and Laporte, 1996). The search space is a bounded domain which contains all possible solutions.

The key research issue in applying metaheuristic-based methods to any combinatorial optimisation problem is to *strike a balance between exploration and exploitation* during the search (Qu et al., 2009). Note that, during the *exploration* stage, the search is encouraged to explore the not-yet-visited search space regions if need be. During the *exploitation* stage, however, the search concentrates on the the already-visited search space regions (Blum and Roli, 2003).

In general, metaheuristics are divided into two categories, *local search-based* and *population-based* methods. Local search-based methods consider *one* solution at a time (Blum and Roli, 2003). The solution undergoes changes iteratively until a static point near the initial solution is reached. Local search-based methods often use neighborhood structures guided by a given acceptance rule to improve the solution. Although the main merit of using these methods is their strength of fine-tuning the solution more structurally and more quickly than population-based methods (Blum and Roli, 2003), the main drawback of those local search-based methods is that they may easily get stuck in local optima. The main cause for the local optimal problem is that local search-based methods focus on exploitation rather than exploration which means that they move in one direction without performing a wider scan of the entire search space. The local search-based methods applied to the course timetabling problem include iterative local search (Socha et al., 2002), simulated annealing (Chiarandini et al., 2006; Kostuch, 2005), very large neighborhood search (Abdullah et al., 2007b, 2005), great deluge (McMullan, 2007; Landa-Silva and Obit, 2008, 2009; Obit et al., 2009; Turabieh et al., 2009).

Population-based methods, which consider *many* solutions at a time, have also been applied to the course timetabling problem. During the search, they recombine current solutions to obtain new ones. Unfortunately, the quality of timetabling solutions produced by population-based methods are inferior to local search-based methods because they are poorer at finding the precise optimal solution in the search space region to which the algorithm converges (Fesanghary et al., 2008). The common cause of this problem is that the population-based methods are more concerned with exploration rather than exploitation. It should be emphasized that population based methods scan the solutions in the entire search space without rigorous concentration on current solutions. The population-based methods applied to the course timetabling problem include Genetic Algorithm (GA) (Lewis and Paechter, 2004, 2005), Ant Colony Optimisation (ACO)(Socha et al., 2002), and Artificial Immune System (AIS) (Malim et al., 2006). Overviews of previous methods for the course timetabling problem are available in the following surveys (Burke et al., 1997; Carter and Laporte, 1997; Burke and Petrovic, 2002; Burke et al., 2004; Lewis, 2008).

The key component (or operator) among local search-based methods has been the *neighborhood structures* (i.e., move, swap, Kempe chain) which are able to explore the search space using one or more local changes in the current solution. On the other hand, the common component among population-based methods has been the *recombination* (i.e., crossover in GA, global-best operator in Particle Swarm Optimisation (PSO), etc.). Recombination exploits the characteristics of the current population in the process of producing a new population. Both local search-based and population-based methods may have a *randomness* (i.e., mutation in GA, cooling schedule in SA) component to diversify the search when and if necessary.

In a recent comprehensive survey of timetabling, Qu et al. (2009) conclude: "There are many research directions generated by considering the *hybridisation* of meta-heuristic methods particularly between population based methods and other approaches". In general, there

are many research trends highlighting the efficiency of using local search-based methods within population-based methods. For example, Blum and Roli (2003) in an influential article on metaheuristics conclude that "In summary, population-based methods are better in identifying promising areas in the search space, whereas trajectory methods are better in exploring promising areas in the search space. Thus, metaheuristic *hybrids* that in some way manage to combine the advantage of population-based methods with the strength of trajectory methods are often very successful".

1.1.4 Harmony Search Algorithm

The Harmony Search Algorithm (HSA) is a new metaheuristic developed by Geem et al. (2001). It mimics the musical improvisation process in which a group of musicians play the pitches of their musical instruments together seeking a pleasing harmony as determined by an audio-aesthetic standard. HSA is a stochastic search mechanism that requires no derivation information in the initial search (Lee et al., 2005). Such method has been successfully applied to a wide variety of optimisation problems (Ingram and Zhang, 2009).

HSA is an iterative improvement method initiated with a number of provisional solutions stored in the 'Harmony Memory (HM)'. At each iteration, a new solution called 'new harmony' is generated based on three operators: (i) 'Memory Consideration', which makes use of accumulative search; (ii) 'Random Consideration', used to diversify the new harmony, and (iii) 'Pitch Adjustment', analogous to local search. A new harmony is then evaluated against an objective function and substituted with the worst harmony stored in HM. This process is repeated until an acceptable solution is obtained.

HSA has an interesting feature that differentiates it from the other metaheuristics: through the Memory consideration and Random consideration, HSA iteratively recombines the charac-

teristics of many solutions in order to generate one solution. It is able to fine tune this solution to which the algorithm converges using pitch adjustment. As such, the HSA has the advantage of combining key components of population-based and local search-based methods in an optimisation model.

1.2 Motivation and Problem Statement

For those with little or no experience in the field, generating a course timetable might be thought of as being trivial. But for the administrator it may be considered a very challenging task because there are a limited number of resources to be assigned to meet a considerable number of requirements. In computing terms, course timetabling is a combinatorial optimisation problem which belongs to the NP-complete class in almost all its variations (Lewis, 2008). Technically, course timetabling is the problem of assigning given courses, each with given features, to given rooms and timeslots according to predefined constraints.

The most successful methods for difficult timetabling problems have been based on meta-heuristics (i.e., population-based and local search-based methods)(Lewis, 2008). Population-based methods are able to explore multiple search space regions at a time. However, they are often poorer at finding a precise local optimal solution in each search space region to which they converge (Fesanghary et al., 2008). In contrast, Local search-based methods are able to fine-tune the search space region to which they converge and find a precise local optimal solution. However, they go through a trajectory in the search space without doing a wider scan of the entire search space. A possible way to tackle the timetabling problem is to strike a balance between global exploration using the strength of population-based methods, and local exploitation using the strength of local search-based methods.

HSA is considered a population-based algorithm with local search-based characteristics

(Lee et al., 2005). However, existing works have not investigated harmony search algorithm in the context of timetabling in general. The focus of the research of this dissertation is to adapt HSA for course timetabling for purposes of unveiling advantages or disadvantages and then enhancing its efficiency by means of combining components from metaheuristics.

1.3 Research Objectives

The aim of this dissertation is not only to develop efficient harmony search-based algorithms for course timetabling problems but also to show that these algorithms can outperform other algorithms in timetabling published in the literature. Thus, new alternatives for solving course timetabling problems are provided.

The key objectives of the research presented in this dissertation are as follows:

1. To adapt the harmony search algorithm for course timetabling.
2. To hybridise the adapted HSA with components from other metaheuristics in order to further improve the quality of the solution.

1.4 Research Contributions

The research has made main contributions to the literature as it has:

1. adapted the HSA to course timetabling and therefore provided the timetabling community with an efficient template for applying HSA to the timetabling problems, henceforth called Basic Harmony Search Algorithm (BHSA).
2. introduced three hybridised versions of the adapted harmony search algorithm with other metaheuristic components. Note that the three versions were introduced sequentially,

each to overcome the weaknesses of the previous one and thus obtain a high-quality solution to UCTP. These can be described as follows:

(a) *Modified Harmony Search Algorithm (MHSA)*. Hybridisation of BHSA with global best concept from Particle Swarm Optimisation (PSO) to improve its speed of convergence. Furthermore, hybridising three neighbourhood structures which were guided by the objective function with BHSA to improve its local exploitation.

(b) *Harmony Search Algorithm with Multi-Pitch Adjusting Rate (HSA-MPAR)*. Hybridisation of BHSA with global best concept from PSO to improve its speed of convergence. Furthermore, hybridising eight different neighbourhood structures which were guided by the objective function with BHSA to increase its ability in local exploitation.

(c) *Hybrid Harmony Search Algorithm (HHSA)*. Hybridisation of BHSA with global best concept from PSO to improve its speed of convergence. Furthermore, hybridising hill climbing optimizer with BHSA as a new operator to increase its ability to find the local optimal solution in the search space of the new harmony solution.

The results obtained were compared with a total of 21 other previous methods in terms of penalty value using the same dataset defined by Socha et al. (2002).

1.5 Overview of Methodology

This section provides a brief discussion on the methodology, described fully in chapter 4, to achieve the research objectives.

For the first objective, the course timetabling data and the way of evaluating each solution with suitable data structures are embedded into BHSA. Although the results were not impres-

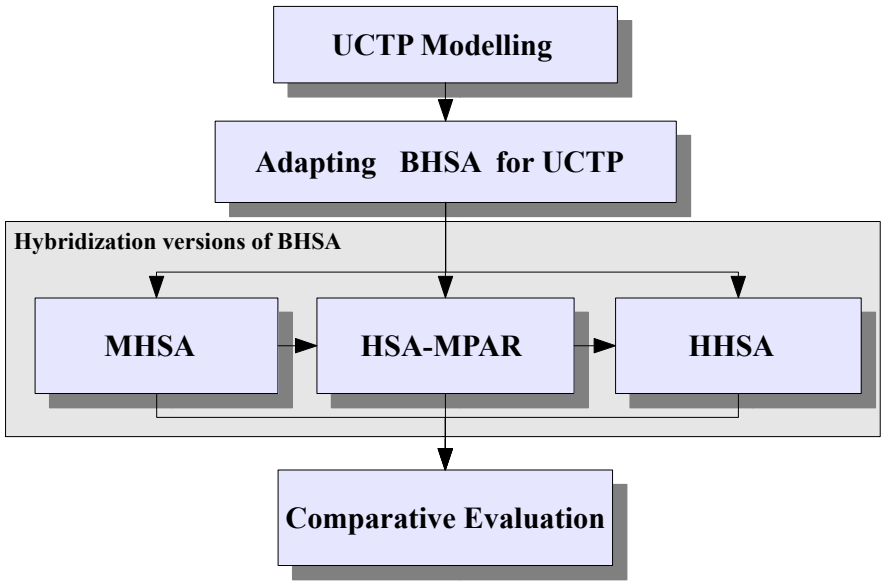

Figure 1.2: Research Methodology

sive, the approach was worthy of further research in order to achieve the second objective.

For the second objective, three hybridisation techniques of effective components from metaheuristics were incorporated into BHSA. The first hybridises the global-best concept from PSO and three neighbourhood structures guided by objective function with BHSA to enhance its convergence speed and local exploitation aspect (i.e., MHSA). The second hybridises the global-best concept from PSO and eight neighbourhood structures guided by objective function with BHSA to further improve local exploitation aspects (i.e., HSA-MPAR). The third hybridises the global-best concept from PSO and hill climbing operator with BHSA to further improve the local exploitation (i.e., HHSA).

For the purpose of evaluating the efficiency of the proposed algorithms, comprehensive comparisons were provided against a total of 21 methods.

Figure 1.2 shows that the stages of the research methodology. Initially, the UCTP is mathematically modeled to be embedded into the BHSA. In the first stage, BHSA is adapted for UCTP to achieve the first objective. Two drawbacks were determined related to convergence

rate and local exploration. The second stage is carried out to overcome the two drawbacks of BHSA by hybridising it with efficient metaheuristic components. This leads to MHSA, HSA-MPAR, and HHSA sequentially. Note that in Figure 1.2, the arrow between the hybridised versions shows that they are sequentially proposed, each to overcome the weaknesses of the previous one.

1.6 Overview of Dissertation

This dissertation includes ten chapters organized as follows: Chapter 2 discusses the basics of the harmony search algorithm. The analogy between musical and optimisation terms are provided.

Chapter 3 provides an overview of the timetabling problems with particular attention to course timetabling. It also surveys some previous methods that tackled the university timetabling problems successfully. The modeling for the course timetabling is described in Chapter 4 which also includes a thorough description of the methodology and the procedures that were conducted.

Chapters 5, 6, 7, 8 present the BHSA, MHSA, HSA-MPAR, HHSA consecutively. It should be noted that each chapter describes a particular proposed method when it is adapted or hybridised for course timetabling and presents the experiments and results with detailed analysis of studying the behaviour of that method in terms of the convergence properties.

In Chapter 9, a comparative analysis of the proposed methods and the previous methods in the literature are provided. Finally in Chapter 10, the research conclusion together with some possible future work are presented.

CHAPTER 2

FUNDAMENTALS TO HARMONY SEARCH ALGORITHM

2.1 Introduction

This chapter provides an explanation to Harmony Search Algorithm (HSA). The characteristics of HSA are provided in Section 2.2. The equivalences between optimisation and musical terms are explained in Section 2.3. Finally the steps of HSA are discussed in Section 2.4. Note that further explanation about the optimisation discipline is provided in Appendix A.

2.2 Harmony Search Algorithm

HSA is a recent metaheuristic population-based method developed by Geem (2000) and Geem et al. (2001). It stems form the musical improvisation process in which a group of musicians play the pitches of their musical instruments seeking for a pleasing harmony as estimated by an audio-aesthetic standard. This is similar to the optimisation process which will be discussed in more detail in Section 2.3.

The HSA is in a sense similar to other metaheuristics in the following characteristics: (Lee and Geem, 2005; Mahdavi and Abolhassani, 2009):

1. HSA is a stochastic search mechanism which does not require derived information in the initial search.

2. In HSA, the objective function guides the search which distinguishes between the poor

and good solutions.

3. HSA is simple to be tailored for a wide variety of optimisation problems.

4. HSA has an iterative improvement procedure called 'improvisation process'. During each iteration, a resulting solution called 'new harmony' is obtained based on operators that can efficiently and effectively vacillate with harmony across the search space.

5. HSA has very few requirements for mathematical computation.

This algorithm has an interesting feature that differentiates it from the other metaheuristics: it iteratively explores the search space by combining multi-search space regions to visit a single search space region. We have to recall that, through the recombination and randomness, the harmony search algorithm iteratively recombines the characteristics of many solutions in order to make one solution. It is able to fine tune this solution to which the algorithm converges using neighborhood structures. Throughout the process recombination is represented by memory consideration, randomness by random consideration, and neighborhood structures by pitch adjustment. In the typical population-based methods, the search space is explored by moving from multi-search space regions to multi-search space regions and the local search-based methods explore the search space regions moving from a single region to another. As such, the harmony search algorithm has the advantage of combining key components of population-based and local search-based methods in an optimisation model.

2.3 Analogy between Musical and Optimisation Contexts

Before providing any explanation, it is worth perusing into Table 2.1 which shows the comparison factors between the optimisation and musical contexts. Figure 2.1 shows the analogy between the music improvisation process and optimisation process. In musical improvisation, a group of musicians improvise the pitches of their musical instruments, *practice after practice*,

Table 2.1: The Optimisation Terms in the Musical Context

Musical Terms		Optimisation Terms
Improvisation	$\longleftrightarrow$	generation or construction
Harmony	$\longleftrightarrow$	Solution vector
Musician	$\longleftrightarrow$	Decision variable
Pitch	$\longleftrightarrow$	Value
Pitch Range	$\longleftrightarrow$	Value Range
audio-aesthetic standard	$\longleftrightarrow$	Objective function
Harmony memory	$\longleftrightarrow$	Memory matrix
Practice	$\longleftrightarrow$	Iteration
Pleasing harmony	$\longleftrightarrow$	(Near-) optimal solution

seeking for a pleasing harmony as determined by an audio-aesthetic standard. Initially, each musician improvises any pitch from the possible pitch range which will finally lead all musicians to make a fresh harmony. That fresh harmony is estimated by an audio-aesthetic standard: if it is good (i.e., involves better pitches than the preferable pitches in musicians' memory), the musicians retain the good pitches in their memory instead of those included within the worst harmony stored earlier for using them in the next practice. *Practice after practice*, the good pitches are stored in the musicians' memory which give them a better chance to produce a pleasing harmony in the following practices.

This can be translated into the optimisation process as follows: a set of decision variables is assigned with values, *iteration by iteration*, seeking for a '*good enough*' solution as evaluated by an objective function. Initially, each decision variable is assigned by any value from its possible range which will finally lead all decision variables to make a new solution vector. That solution vector is evaluated by an objective function: if it is good (i.e., involves better values than experience values stored in the memory), the decision variables will store the good values in their memory instead of those included within the worst solution vector stored earlier for using them in the next iterations. *Iteration by iteration*, the good values for each decision variable will be stored in the memory giving them a better chance to produce a better solution in the following iterations.

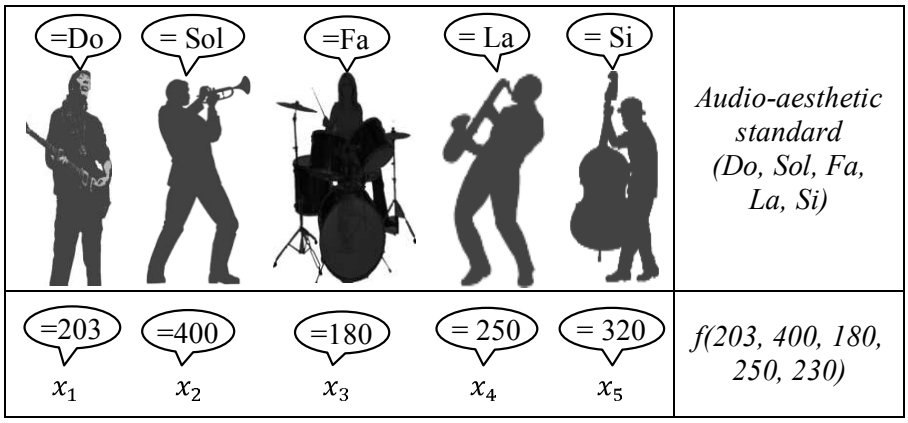

Figure 2.1: Analogy between Music Improvisation and Optimisation Process

When each musician improvises a pitch from his musical instrument, there are three options: (i) improvising any pitch from his memory, (ii) modifying a pitch which exists in memory, or (iii) improvising a pitch from the possible pitch range. Analogously, in the optimisation context, the value of each decision variable is decided according to one of the following options: (i) assigning a value stored in the memory, (ii) modifying a value which exists in the memory, or (iii) assigning a value from its feasible range. Geem et al. (2001) formalized these three options into three operators: memory consideration, pitch adjustment, and random consideration (those will be discussed in more detail in Section 2.4).

Figure 2.2 shows the harmony memory structure which is the core part of the improvisation process. Consider musical instruments of five musicians on the Jazz bandstand. They have sets of preferable pitches in their memory as follows: Guitarist: $\{Do, Mi, Sol\}$; Trumpeter: $\{La, Si, Sol\}$; Drummer:$\{Re, Sol, Si\}$; Saxophonist: $\{ Fa, Do, La\}$; Double bassist: $\{Re, Sol, Mi\}$. Assume in a practice if Guitarist randomly improvises $\{Do\}$ from his memory; Trumpeter improvises $\{Sol\}$ from his memory; Drummer adjusts $\{Re\}$ from his memory to $\{Fa\}$, Saxophonist improvises $\{La\}$ from his memory, and Double bassist improvises $\{Si\}$ from the available range $\{Do, Re, Mi, Fa, Sol, Si\}$. All these pitches together create a new harmony (Do, Sol, Fa, La, Si) which is estimated by an audio-aesthetic standard. If the new

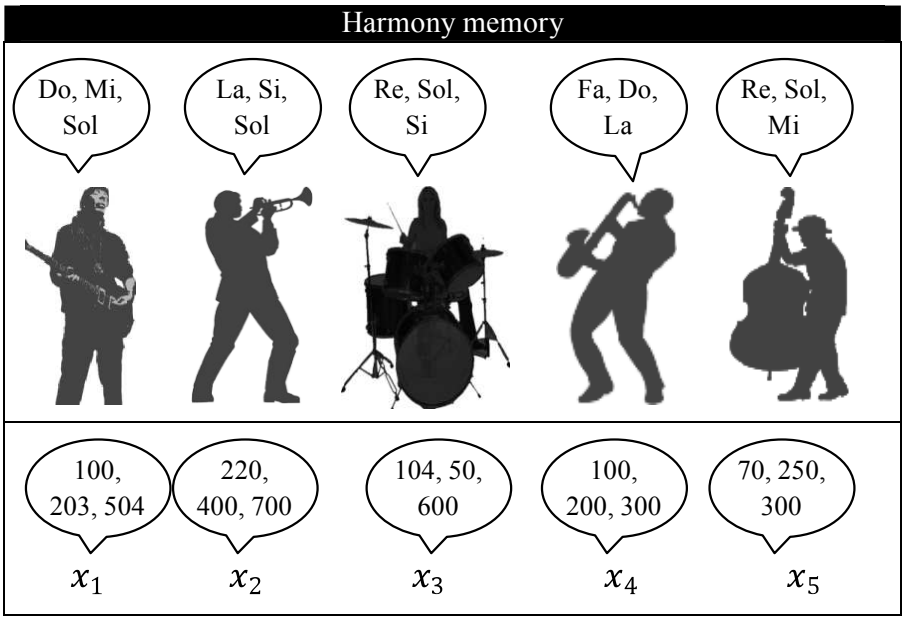

Figure 2.2: The Harmony Memory Structure

harmony is better than the worst harmony stored in the harmony memory, the worst harmony existing in harmony memory is substituted with the new one. This process is repeated until a pleasing harmony is obtained.

In terms of optimisation: consider five decision variables, each of which has stored experience values in the harmony memory as follows: x_1 :$\{100,203,504\}$; x_2 : $\{220,400,700\}$; x_3 :$\{104,50,600\}$; x_4 :$\{100,200,300\}$; x_5 :$\{70,250,300\}$. Suppose in an iteration, if x_1 is assigned with 203 from its memory; x_2 is assigned with 400 from its memory; x_3 is adjusted from the value 104 stored in its memory to be 180; x_4 is assigned with 200 from its memory; x_5 is assigned with 320 from its feasible range $x_3 \in [0,600]$. The new harmony solution $(203,400,180,200,320)$ is evaluated by an objective function. If the new harmony solution is better than the worst solution in the harmony memory, then it replaces the worst solution. This process is repeated until an optimal solution is considered to have been reached.

2.4 Harmony Search Procedure

Algorithm 2.1 shows the pseudo-code of the classical HSA with five main steps that will be described as follows:

Algorithm 2.1 The classical harmony search algorithm

STEP1 Initialise the problem and the HSA parameters

1: Input data instance of the optimisation problem
2: Set the HSA parameters (HMCR, PAR, NI, HMS).

STEP2 Initialise the harmony memory

1: Construct vectors of the harmony memory, $\mathbf{HM} = \{\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^{\mathrm{HMS}}\}$
2: Recognise the worst vector in $\mathbf{HM}$, $\boldsymbol{x}^{\mathrm{worst}} \in \{\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^{\mathrm{HMS}}\}$

STEP3 Improvise a new harmony

1: $\boldsymbol{x}' = \phi$ /**$\boldsymbol{x}'$ is the new harmony vector**/
2: **for** $i = 1, \cdots, N$ **do**
3: **if** $(U(0,1) \leq \mathrm{HMCR})$ **then**
4: $x_i' \in \{x_i^1, x_i^2, \ldots, x_i^{\mathrm{HMS}}\}$ /**memory consideration**/
5: **if** $(U(0,1) \leq \mathrm{PAR})$ **then**
6: $x_i' = v_{i,k\pm m}$ /** pitch adjustment **/
7: **end if**
8: **else**
9: $x_i' \in \boldsymbol{X}_i$ /** random consideration **/
10: **end if**
11: **end for**

STEP4 Update the harmony memory (HM)

1: **if** $(f(\boldsymbol{x}') < f(\boldsymbol{x}^{\mathrm{worst}}))$ **then**
2: Include $\boldsymbol{x}'$ to the **HM**.
3: Exclude $\boldsymbol{x}^{\mathrm{worst}}$ from **HM**.
4: **end if**

STEP5 Check the stop criterion

1: **while** (not termination criterion specified by NI is met) **do**
2: Repeat **STEP3** and **STEP4**
3: **end while**

Step 1 *Initialise the problem and the HSA parameters.* Suppose that the discrete optimisation

problem is modeled as in Eq.(2.1).

$$\min\{f(\boldsymbol{x})|\boldsymbol{x}\in\mathbf{X}\} \quad \text{Subject to} \quad g(\boldsymbol{x})<0 \quad \text{and} \quad h(\boldsymbol{x})=0 \tag{2.1}$$

Where $f(\boldsymbol{x})$ is the objective function; $\boldsymbol{x}=\{x_i|i=1,\ldots,N\}$ is the set of each decision variable. $\mathbf{X}=\{\boldsymbol{X}_i|i=1,\ldots,N\}$ contains all possible *discrete* values for each decision variable, i.e., $\boldsymbol{X}_i=\{v_{i,1},v_{i,2},\ldots,v_{i,K_i}\}$. N is the number of decision variables, and K_i is the number of values for each decision variable x_i. $g(\boldsymbol{x})$ are inequality constraint functions and $h(\boldsymbol{x})$ are equality constraint functions. The parameters of the HSA required to solve the optimisation problem are also specified in this step:

(a) The Harmony Memory Consideration Rate (HMCR) is used in the improvisation process to determine whether the value of a decision variable is to be selected from the solutions stored in the Harmony Memory (HM).

(b) The Harmony Memory Size (HMS) is similar to the population size in genetic algorithm.

(c) The Pitch Adjustment Rate (PAR) decides whether the decision variables are to be adjusted to a neighbouring value.

(d) The Number of Improvisations (NI) corresponds to the number of iterations.

Note that the HMCR and PAR are the two parameters responsible for the improvisation process. These parameters will be explained in more detail in the next steps.

Step 2 *Initialise the harmony memory*. The harmony memory (HM) is a memory location which contains sets of solution vectors which are determined by HMS (see Eq.2.2). In this step, these vectors are randomly (or *heuristically*) constructed and stored to the HM

based on the objective function values.

$$\mathbf{HM} = \begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_N^1 \\ x_1^2 & x_2^2 & \cdots & x_N^2 \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{\text{HMS}} & x_2^{\text{HMS}} & \cdots & x_N^{\text{HMS}} \end{bmatrix} \tag{2.2}$$

Step 3 *Improvise a new harmony*. In this step, the HSA will generate (*improvise*) a new harmony vector from scratch, $\boldsymbol{x}' = (x_1', x_2', \cdots, x_N')$, based on three operators: (1) memory consideration, (2) random consideration, and (3) pitch adjustment.

Memory consideration. In memory consideration, the value of the first decision variable x_1' is randomly assigned from the historical values, $\{x_1^1, x_1^2, \ldots, x_1^{HMS}\}$, stored in HM vectors. Values of the other decision variables, $(x_2', x_3', \ldots, x_N')$, are sequentially assigned in the same manner with probability of HMCR where $0 \leq \text{HMCR} \leq 1$. The operation of this operator is similar to the recombination operator in other population-based methods and is a good source of exploitation (Yang, 2009).

Random consideration. Decision variables that are not assigned with values according to memory consideration are randomly assigned according to their possible range by random consideration, with a probability of (1-HMCR) as in Eq.(2.3).

$$x_i' \leftarrow \begin{cases} x_i' \in \{x_i^1, x_i^2, \ldots, x_i^{HMS}\} & \text{w.p.} \quad \text{HMCR} \\ x_i' \in \boldsymbol{X}_i & \text{w.p.} \quad \text{1 - HMCR} \end{cases} \tag{2.3}$$

Random consideration is functionally similar to the mutation operator in genetic algorithm which is a source of global exploration in HSA (Yang, 2009). The HMCR parameter is the probability of assigning one value of a decision variable, x_i', based on historical values stored in the HM. For instance, if (HMCR =0.90), this means that

the probability of assigning the value of each decision variable from historical values stored in the HM vectors is with the probability of 90%, and the value of each decision variable is assigned from its possible value range is with the probability of 10%.

Pitch adjustment. Every decision variable x_i' of a new harmony vector, $\boldsymbol{x}' = (x_1', x_2', x_3', \ldots, x_N')$, that has been assigned a value by memory considerations is examined for whether or not it should be pitch adjusted with the probability of PAR where $0 \leq PAR \leq 1$ as in Eq.(2.4).

$$\text{Pitch adjusting decision for } x_i' \leftarrow \begin{cases} \text{Yes} & \text{w.p.} \quad \text{PAR} \\ \text{No} & \text{w.p.} \quad \text{1-PAR} \end{cases} \tag{2.4}$$

A PAR of 0.10 means that the HSA modifies the existing value of decision variables assigned by memory consideration with a probability of $(\text{PAR} \times \text{HMCR})$, while the other values of decision variables assigned by memory consideration do not change. If the pitch adjustment decision for x_i' is Yes, the value of x_i' is modified to its neighboring value as follows:

$$x_i'(k) = v_{i,k \pm m} \tag{2.5}$$

where x_i' is assigned with value $v_{i,k}$, that is, the kth element in $\boldsymbol{X}_i$. m is the neighboring index, $m \in \mathbb{Z}$. The following summarizes the improvisation process of step 3, which is the main mechanism for iterating towards an optimal solution:

$$x_i' \leftarrow \begin{cases} x_i' \in \{x_i^1, x_i^2, \ldots, x_i^{\text{HMS}}\} & \text{w.p.} \quad \text{HMCR} \\ x_i' = v_{i,k \pm m} & \text{w.p.} \quad \text{HMCR} \times \text{PAR} \\ x_i' \in \boldsymbol{X}_i & \text{w.p.} \quad \text{1 - HMCR} \end{cases} \tag{2.6}$$

It is clear that the HSA assumes implicitly that good harmony vectors consist of good

harmony decision variables. It also assumes that the process of learning the harmony memory by means of storing good values for each decision variable can help to produce a good quality new harmony.

Step 4 *Update the harmony memory*. If the new harmony vector, $\boldsymbol{x}' = (x'_1, x'_2, \cdots, x'_N)$, is better than the worst harmony stored in HM in terms of the objective function value, the new harmony vector replaces the worst harmony vector. Basically, each update of HM causes HSA to focus on the search space regions containing high quality solutions.

Step 5 *Check the stop criterion*. Step 3 and step 4 of HSA are repeated until the stop criterion (maximum number of improvisation) is met. This is specified by NI parameter.

2.5 Conclusion

HSA is a recent metaheuristic population-based method. This algorithm has been successfully applied to a wide variety of optimisation problems as shown in Appendix B. The concepts of HSA are described within the context of creating a pleasing harmony within a musical context. Finally, the main steps of HSA have been thoroughly described. Note that further explanation about other optimisation methods has been provided in Appendix A.

CHAPTER 3

LITERATURE REVIEW

3.1 Introduction

This chapter will discuss the timetabling problems in general and university course timetabling problem (UCTP) in particular. The methods tackling UCTP will be surveyed and eventually a comprehensive analysis to the exiting methods will be conducted.

3.2 Timetabling Problems

A comprehensive definition for timetabling problems was put forward by Burke et al. (2004), "*A timetabling problem is a problem with four parameters: T, a finite set of times; R, a finite set of resources; M, a finite set of meetings; and C, a finite set of constraints. The problem is to assign times and resources to the meetings so as to satisfy the constraints as far as possible.*"

Timetabling problems appeared in various forms with reference to different disciplines including Educational Timetabling (Lewis, 2008; Carter and Laporte, 1997; Schaerf, 1999; Burke et al., 1997; Qu et al., 2009; Burke et al., 2004; Burke and Petrovic, 2002), Nurse Rostering (Heang et al., 2003), Transportation Timetabling (Wren and Rousseau, 1995), Employee Timetabling (Meisels and Lusternik, 1998) and Sport Scheduling (Wright, 2007).

Educational timetabling problems in particular have been studied intensively over five decade. They comprise three forms (Schaerf, 1999): (i) school timetabling problems, (ii) course timetabling problems, and (iii) examination timetabling problems.

Table 3.1: Differences and Similarities between Examination and Course Timetabling

	Course Timetabling	Examination Timetabling
Similarities	– It is the process of scheduling a set of courses to rooms and timeslots. – It is NP-complete in often all their variations. – The constraints are related to the preferences of lecturers and students.	– It is the process of assigning exams to rooms and timeslots. – It is NP-complete in often all their variations. – The constraints are related to the preferences of lecturers and students
Differences	– One course can be timetabled in the same room at the same timeslot. – Weekly-based. The course timetable is designed for one week and repeated for others (the number of timeslots required is fixed).	– Many exams can be timetabled in the same room at the same timeslot. – Periodic-based. The number of timeslots is predetermined or should be minimised (number of timeslots required is fixable in most cases).

University timetabling involves two forms of educational timetabling problems: course and examination timetabling problems. Although they are solved separately, they share some characteristics as shown in Table 3.1. Lewis (2008) clarified the similarities between both problems while McCollum (2006) explained the major differences in each research domain. Generally speaking, university timetabling is the process of assigning a set of events (e.g., courses, examinations, tutorials), each with a set of features, to a set of rooms and timeslots according to a set of constraints.

Conventionally, the constraints are divided into two types (Burke et al., 1997):

1. **Hard Constraints**. This type of constraints must be essentially satisfied in the timetabling solution to be *feasible*.

2. **Soft Constraints**: The satisfaction of soft constraints in the timetabling solution is not essential but desirable. The quality of timetabling solution is normally determined against the number of the violations of the soft constraints. For example, if the quality of

Table 3.2: The Timetabling Constraint Classes (Lewis, 2008)

Constraint Class	Descriptions
Unary constraints	A constraint involving one event. For example, an event has to be scheduled in specific timeslot and room.
Binary constraints	A constraint involving two events. For example, two events that share the same student cannot be scheduled in the same timeslot. This class of constraints is called '*event clash*'.
Capacity constraints	A constraint related to room capacity. For example, the event must be scheduled in a room with sufficient capacity.
Event spread constraints	A constraint related to the time gap between events that share common resources. For example, the student may prefer to have sufficient free time before the exam dates so that he can revise his exams. To cite another example, a student may not wish to attend three events in a row.
Agent constraints	A constraint related to personal preferences. For example, a lecturer may prefer to teach a particular event or may like to be free on a Friday morning.

a timetabling solution is 20, this means that there are 20 violations in the soft constraints (assuming that each violation in any soft constraint has a similar weight equal to 1).

Corne et al. (1995) distinguished between five general classes of university timetabling constraints (See Table 3.2). A common constraint on the university timetabling problems is the *'event clash'* where the events that share the same student must be scheduled in a different timeslot (Lewis, 2008). A comprehensive list of constraints in the literature can be seen in (Pongcharoen et al., 2008).

3.3 Graph Colouring Model for University Timetabling

The basic timetabling problems[1] can be modeled as a graph colouring (Welsh and Powell, 1967; Wood, 1969; Werra, 1985). A graph $G = (V, E)$ is an indirect graph with a set of N vertices

[1] Basic timetabling forms include the process of assigning events to timeslots only but without rooms.