



Faculty of Computer Science and Information Technology

**Improved Variable Neighbourhood Descent (VND) to Solve Universiti
Malaysia Sarawak (UNIMAS) Course Timetabling Problem**

Chen Mei Ching

**Doctor of Philosophy
2023**

Improved Variable Neighbourhood Descent (VND) to Solve Universiti
Malaysia Sarawak (UNIMAS) Course Timetabling Problem

Chen Mei Ching

A thesis submitted

In fulfillment of the requirements for the degree of Doctor of Philosophy

(Artificial Intelligence)

Faculty of Computer Science and Information Technology

UNIVERSITI MALAYSIA SARAWAK

2023

DECLARATION

I declare that the work in this thesis was carried out in accordance with the regulations of Universiti Malaysia Sarawak. Except where due acknowledgements have been made, the work is that of the author alone. The thesis has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.



.....

Signature

Name: Chen Mei Ching

Matric No.: 19010037

Faculty of Computer Science and Information Technology

Universiti Malaysia Sarawak

Date : 22 August 2023

ACKNOWLEDGEMENT

I would like to sincerely thank everyone who has contributed to this thesis, whether directly or indirectly.

First of all, I would like to express my thanks to my family members for giving me the support and motivation to pursue my PhD. They gave me the strength to keep pursuing my research journey.

My sincere gratitude goes to my supervisor, AP Dr Sze San Nah, and my co-supervisors, Dr Goh Say Leng and AP Dr Chiew Kang Leng, for their guidance and support in completing this study. Their valuable knowledge has been the greatest input in helping me to conduct my research. They have sacrificed their valuable time to meet me and help me during my PhD journey by giving many suggestions to improve my research output.

I also would like to express my gratitude to my friends who are willing to become channels for me to release my stress. They are willing to sacrifice their time to listen to me and be one of the outlets for me to release my emotions.

I also wish to express my appreciation to the Faculty of Computer Science and Information Technology for providing the facilities needed to do my research. My sincere thanks to the Faculty of Computer Science and Information Technology administrator/timetable planner for providing the real-world dataset needed for my research.

My sincere gratitude to the Centre for Graduate Studies for the advice and support given during my period of study at Universiti Malaysia Sarawak. Finally, I would like to thank the

management of the Universiti Malaysia Sarawak for making it possible for me to complete my study here in Sarawak. Thank you all.

ABSTRACT

Academic institutions face the timetabling problem every semester. The task of allocating lectures to limited timeslots and venues must fulfil certain constraints unique to each educational institution. This study investigated heuristic orderings and the variable neighbourhood descent approach to tackle the course timetabling problem at the Faculty of Computer Science and Information Technology (FCSIT), Universiti Malaysia Sarawak (UNIMAS) on the basis of the students. The objectives of the study were to formulate a mathematical model and improve a computational bounded heuristics-based solution to solve the course timetabling problem at the faculty. A two-stage heuristic algorithm is proposed. In stage 1, heuristic orderings were utilised to find a feasible solution using 31 timeslots instead of the 48 timeslots in the existing timetabling software. In stage 2, the variable neighbourhood descent approach with new neighbourhood structures was utilised to improve the quality of the solution. The improved algorithm was tested on real-world data instances (in semesters 1 and 2 of 2019/2020) at the FCSIT, UNIMAS. The results show that certain heuristic orderings (the largest degree or the combination of the largest degree and largest enrolment in descending order) are better than others in generating a feasible solution. In stage 2, the proposed algorithm with new neighbourhood structures managed to reduce the soft constraint violations for instances in semesters 1 and 2. Sensitivity analysis was performed on the proposed algorithm. The experimental results demonstrate the flexibility of the proposed algorithm in solving the university course timetabling problem (UCTTP) at the FCSIT.

Keywords: University course timetabling, optimisation, heuristic orderings, variable neighbourhood descent, perturbation

***Penambahbaikan Penurunan Pembolehubah Kejiranan Untuk Menyelesaikan
Masalah Penjadualan Kursus Universiti Malaysia Sarawak (UNIMAS)***

ABSTRAK

Institusi akademik berhadapan dengan masalah penjadualan pada setiap semester. Tugas memperuntukkan kuliah ke slot masa dan tempat yang terhad mesti memenuhi kekangan tertentu yang unik bagi setiap institusi pendidikan. Dalam kajian ini, kaedah penyusunan heuristik dan penurunan pembolehubah kejiranan dalam menangani masalah penjadualan kursus di Fakulti Sains Komputer dan Teknologi Maklumat (FSKTM), Universiti Malaysia Sarawak (UNIMAS) dengan berasaskan pelajar disiasat. Objektif kajian adalah untuk menghasilkan model matematik dan membangunkan satu kaedah heuristik dengan komputasi terhad untuk menyelesaikan masalah penjadualan kursus di fakulti. Heuristik dengan dua fasa telah dicadangkan. Pada fasa 1, penyusunan heuristik digunakan untuk mencari jadual waktu yang sah menggunakan 31 slot masa dan bukannya 48 slot masa seperti dalam perisian penjadualan yang sedia ada. Pada fasa 2, kaedah penurunan pembolehubah kejiranan dengan struktur kejiranan yang baru digunakan untuk meningkatkan kualiti jadual waktu. Algoritma yang dicadangkan diuji dengan menggunakan data sebenar iaitu semester 1 dan 2 tahun akademik 2019/2020 dari FSKTM, UNIMAS. Keputusan menunjukkan kaedah penyusunan heuristik (darjah terbesar dan kombinasi darjah terbesar dan pendaftaran terbesar) adalah lebih baik daripada yang lain dalam menjana jadual waktu yang sah. Pada fasa 2, struktur kejiranan yang baru berjaya meningkatkan kualiti jadual waktu untuk semester 1 dan 2. Analisis sensitiviti dilakukan pada algoritma yang dicadangkan. Keputusan eksperimen menunjukkan fleksibiliti algoritma yang dicadangkan dalam penyelesaian masalah penjadualan kursus universiti di FSKTM.

Kata kunci: *Penjadualan kursus universiti, pengoptimuman, penyusunan heuristik, penurunan pembolehubah kejituan, gangguan*

TABLE OF CONTENTS

	Page
DECLARATION	i
ACKNOWLEDGEMENT	ii
ABSTRACT	iv
<i>ABSTRAK</i>	v
TABLE OF CONTENTS	vii
LIST OF TABLES	xiv
LIST OF FIGURES	xvii
LIST OF ABBREVIATIONS	xix
CHAPTER 1: INTRODUCTION	20
1.1 Study Background	20
1.2 Problem Statement	22
1.3 Objectives	23
1.4 Research Scope	24
1.5 Research Methodology	25
1.6 Research Significance	27
1.7 Thesis Outline	27

CHAPTER 2: LITERATURE REVIEW	30
2.1 Introduction	30
2.2 Approaches/ Methodologies in Benchmark UCTTP	30
2.2.1 Operational Research (OR) Based Techniques	31
2.2.2 Single Solution-based Meta-heuristics	33
2.2.3 Population-based Meta-heuristics	36
2.2.4 Hyper-heuristic	40
2.2.5 Multi Criteria/ Objective Approaches	40
2.2.6 Hybrid Approaches	40
2.3 Approaches/ Methodologies in Real-world UCTTP	42
2.3.1 Operational Research (OR) Based Techniques	42
2.3.2 Single Solution-based Meta-heuristics	42
2.3.3 Population-based Meta-heuristics	44
2.3.4 Hyper-heuristic	45
2.3.5 Hybrid Approaches	45
2.4 Perspectives in UCTTP	48
2.5 Trends in Benchmark UCTTP	51
2.5.1 Socha Benchmark Dataset	53
2.5.2 ITC-02 Benchmark Dataset	53
2.5.3 ITC-07 (Track 2) Benchmark Dataset	54

2.5.4	ITC-07 (Track 3) Benchmark Dataset	55
2.5.5	Hard Benchmark Dataset	56
2.5.6	ITC-2019 Benchmark Dataset	58
2.6	Algorithm for the Approaches/methodologies in UCTTP	59
2.6.1	Variable Neighbourhood Descent	59
2.6.2	Tabu Search	62
2.6.3	Simulated Annealing	62
2.6.4	Genetic Algorithm	63
2.7	Limitations of the Approaches/methodologies in UCTTP	64
2.8	Research Opportunities in UCTTP	66
2.9	Research Gap Analysis	68
2.10	Conclusion	69
	CHAPTER 3: CASE STUDY	70
3.1	Introduction	70
3.2	Problem Description	70
3.2.1	Curriculum-based Course Timetabling	71
3.2.2	Timeslots	72
3.2.3	Venues	73
3.2.4	Pre-registration Data Practice	75
3.2.5	Size of Problem Instance	75

3.3	Modelling and Formulation	76
3.3.1	Entities	76
3.3.2	List of Hard Constraints and Soft Constraints	77
3.3.3	Notations	78
3.4	Conclusion	84
CHAPTER 4: PROPOSED ALGORITHM: HEURISTIC ORDERINGS AND		
VARIABLE NEIGHBOURHOOD DESCENT		85
4.1	Introduction	85
4.2	Framework for Two-stage Heuristic Algorithm	85
4.3	Stage 1: Heuristic Ordering (HO)	86
4.3.1	Algorithm Description	86
4.4	Stage 1: Heuristic Ordering (HO) with a Perturbation	88
4.4.1	Algorithm Description	89
4.5	Stage 1: Tabu Search (TS)	93
4.5.1	Algorithm Description	93
4.6	Stage 2: Variable Neighbourhood Descent (VND)	96
4.6.1	Algorithm Description	96
4.7	Stage 2: Variable Neighbourhood Descent (VND) with a Perturbation	104
4.7.1	Algorithm Description	104
4.8	Stage 2: Tabu Search (TS) – Used as a Comparison to the Improved VND	105

4.8.1	Algorithm Description	106
4.9	Conclusion	108
CHAPTER 5: REAL-WORLD UNIVERSITY COURSE TIMETABLING		
	PROBLEM	110
5.1	Introduction	110
5.2	Dataset Description	111
5.3	Stage 1: Heuristic Ordering (HO)	111
5.3.1	Experimental Results and Discussion	111
5.4	Stage 1: HO_P	113
5.4.1	Experimental Results and Discussion	113
5.5	Stage 2: Variable Neighbourhood Descent (VND)	114
5.5.1	Experimental Results and Discussion	114
5.6	Stage 2: Variable Neighbourhood Descent (VND) with a Perturbation	121
5.6.1	Experimental Results and Discussion	121
5.7	Sensitivity Analysis	125
5.7.1	Number of Events	125
5.7.2	Timetable Availability and Large Venue Resources Availability	132
5.7.3	Number of Students	136
5.8	Comparison Between VND_Proposed and Tabu Search (TS)	139
5.8.1	Experimental Results and Discussion	139

5.9	Conclusion	141
CHAPTER 6: ITC-2007 (TRACK 2) BENCHMARK DATASET		144
6.1	Introduction	144
6.2	Problem Description	144
6.3	Dataset Description	146
6.4	Stage 1: Heuristic Ordering with a Perturbation (HO_P)	147
6.4.1	Experimental Results and Discussion	147
6.5	Stage 1: Tabu Search (TS)	148
6.5.1	Experimental Results and Discussion	148
6.6	Stage 2: Variable Neighbourhood Descent (VND)	150
6.6.1	Experimental Results and Discussion	150
6.7	Stage 2: Variable Neighbourhood Descent (VND) with a Perturbation	152
6.7.1	Experimental Results and Discussion	152
6.8	Comparison Between VND_Proposed and Tabu Search (TS)	153
6.8.1	Experimental Results and Discussion	153
6.9	Conclusion	155
CHAPTER 7: CONCLUSION AND RECOMMENDATIONS		156
7.1	Introduction	156
7.2	Conclusion of Research	156
7.3	Research Contributions	160

7.4	Future Work	161
	REFERENCES	162
	APPENDICES	172

LIST OF TABLES

	Page
Table 2.1 Approaches in Solving UCTTP (Benchmark and Real-world Problem Datasets)	48
Table 2.2 Summary of Approaches in UCTTP	51
Table 2.3 Benchmark Datasets and Their Respective State-Of-The-Art Methodologies	52
Table 2.4 The Features of the Socha Benchmark Dataset	53
Table 2.5 The Features of the ITC-02 Benchmark Dataset	54
Table 2.6 The Features of the ITC-07 (Track 2) Benchmark Dataset	55
Table 2.7 The Features of the ITC-07 (Track 3) Benchmark Dataset	56
Table 2.8 The Features of Hard Benchmark Dataset	57
Table 2.9 The Features of the ITC-2019 Benchmark Dataset	58
Table 3.1 FCSIT Data Instances for Academic Years 2019/2020	70
Table 3.2 Usage of Blocked Timeslots	73
Table 3.3 Teaching Venues and their Capacity	74
Table 3.4 Set Notations of the Mathematical Model	78
Table 3.5 Parameter Notations of the Mathematical Model	79
Table 5.1 FCSIT Data Instances for Academic Years 2019/2020	111
Table 5.2 Number of Unallocated Courses for Initial Solutions Generated using Different HO in Stage 1. $N = 30$ runs.	112
Table 5.3 Comparison of the Number of Unallocated Courses using HO and HO_P in Stage 1 for Semester 1's instance. $N = 30$ runs.	113
Table 5.4 Comparison of Number of Unallocated Courses using HO and HO_P in Stage 1 for Semester 2's instance. $N = 30$ runs.	114
Table 5.5 Comparison Between the Number of Soft Constraint Violations that occurred using VND_Proposed and the Existing FCSIT Solution. $N = 30$ runs.	117

Table 5.6	Summary of Soft Constraint Violations for the New NS (NS3 and NS7) in Stage 2 for both instances (semesters 1 and 2) and their Paired Samples <i>t</i> -test Results. <i>N</i> = 30 runs.	120
Table 5.7	Summary of Soft Constraint Violations for the New NS (NS4 and NS8) in Stage 2 for Both Instances (semesters 1 and 2) and their Paired Samples <i>t</i> -test Results. <i>N</i> = 30 runs.	121
Table 5.8	Comparison of Soft Constraint Violations between VND_Proposed and VND_Proposed_P in Stage 2 for Both Instances (Semesters 1 and 2) and their Paired Samples <i>t</i> -test Results. <i>N</i> = 30 runs.	124
Table 5.9	Comparison of Soft Constraint Violations between VND_Proposed, VND_Proposed_P and Existing FCSIT Solution. <i>N</i> = 30 runs.	125
Table 5.10	Feature of Two Extra Events Added to the Database for Semester 1's instance for Sensitivity Analysis	126
Table 5.11	Comparison of the Feature for Semester 1's Instance Before and After the Extra Two Events are Added	126
Table 5.12	Number of Unallocated Courses for Initial Solutions Generated Using Different HO in Stage 1 After Two Extra Events are Added to Semester 1's Instance. <i>N</i> = 30 runs.	127
Table 5.13	Comparison of Number of Unallocated Courses for Initial Solutions Generated Using HO and HO_P in Stage 1 for Semester 1's Instance After the Extra Two Events are Added. <i>N</i> = 30 Runs.	128
Table 5.14	Comparison of Number of Unallocated Courses for Semester 1's Instance (Original) and Semester 1's Instance (Extra Two Events) in Stage 1 for HO and HO_P. <i>N</i> = 30 runs.	129
Table 5.15	Feature of Five Extra Events Added to The Database for Semester 1's Instance for Sensitivity Analysis	129
Table 5.16	Comparison of the Feature for Semester 1's Instance Before and After the Extra Five Events are Added	130
Table 5.17	Number of Unallocated Courses for Initial Solutions Generated Using Different HO in Stage 1 After Five Extra Events are Added to Semester 1's Instance. <i>N</i> = 30 runs.	131
Table 5.18	Comparison of Number of Unallocated Courses for Initial Solutions Generated Using HO and HO_P in Stage 1 for Semester 1's Instance After the Extra Five Events are Added. <i>N</i> = 30 runs.	132
Table 5.19	Comparison of Number of Unallocated Courses for Semester 1's Instance (Original) and Semester 1's Instance (Extra Five Events) in Stage 1 for HO and HO_P. <i>N</i> = 30 runs.	132

Table 5.20	Comparison of Number of Unallocated Courses for Semester 1's Instance (Extra Five Events) in Stage 1 for HO and TS with 31 Timeslots. $N = 30$ runs.	133
Table 5.21	Comparison of Number of Unallocated Courses for Semester 1's Instance (Extra Five Events) in Stage 1 for HO When Available Timeslots are Increased from 31 Timeslots to 34 Timeslots. $N = 30$ runs.	135
Table 5.22	Comparison of Number of Unallocated Courses for Semester 1's Instance (Extra Five Events) in Stage 1 for HO_P When Available Timeslots are Increased from 31 Timeslots to 34 Timeslots. $N = 30$ runs.	135
Table 5.23	Comparison of the Feature for Semester 1's Instance Before and After the Extra Students (100 Students) are Added	137
Table 5.24	Number of Unallocated Courses for Initial Solutions Generated Using Different HO in Stage 1 After 100 Extra Students are Added to Semester 1's Instance. $N = 30$ runs.	138
Table 5.25	Comparison of Number of Unallocated Courses for Initial Solutions Generated Using HO and HO_P in Stage 1 for Semester 1's Instance After the Extra 100 Students are Added. $N = 30$ runs.	139
Table 5.26	Comparison of Number of Unallocated Courses for Semester 1's Instance (Original) and Semester 1's Instance (Extra 100 Students) in Stage 1 for HO and HO_P. $N = 30$ runs.	139
Table 6.1	Number of Timeslots Used in ITC-2007 (Track 2)	145
Table 6.2	The Features of the ITC-2007 (Track 2) Benchmark Dataset.	146
Table 6.3	Comparison of Number of Unallocated Courses using HO and HO_P in Stage 1 for ITC-2007 (Track 2) Instances. $N = 30$ runs.	147
Table 6.4	Comparison of Number of Unallocated Courses using HO_P and HO_P_TS in Stage 1 for ITC-2007 (Track 2) Instances. $N = 30$ runs.	149
Table 6.5	Comparison of Soft Constraint Violations using VND_5 6 and the Proposed VND_5 6 7 8 for ITC-2007 (Track 2) Instances. $N = 30$ runs.	151
Table 6.6	Comparison of Soft Constraint Violations between VND_5 6 7 8 and VND_5 6 7 8_P in Stage 2 for ITC-2007 (Track 2). $N = 30$ runs.	152
Table 6.7	Comparison of Soft Constraint Violations between VND_Proposed and Tabu Search. $N = 30$ runs.	154

LIST OF FIGURES

	Page
Figure 2.1 Neighbourhood Structures	43
Figure 2.2 Case Studies of the Real-world UCTTP	47
Figure 2.3 Classification of Approaches in Benchmark and Real-world UCTTP	50
Figure 2.4 Variable Neighbourhood Descent Algorithm	61
Figure 2.5 Tabu Search Algorithm	62
Figure 2.6 Simulated Annealing Algorithm	63
Figure 2.7 Genetic Algorithm	64
Figure 3.1 Number of Timeslots	73
Figure 4.1 General framework for solving the UCTTP at the FCSIT, UNIMAS	86
Figure 4.2 HO Algorithm	88
Figure 4.3 HO Algorithm with a Perturbation	90
Figure 4.4 Perturbation Procedure for HO Algorithm	92
Figure 4.5 Perturbation Procedure for HO Algorithm in Diagram	93
Figure 4.6 TS Algorithm	95
Figure 4.7 Neighbourhood structures: NS1 to NS8	100
Figure 4.8 VND Algorithm Solving <i>SC1</i>	102
Figure 4.9 VND Algorithm Solving <i>SC2</i> and <i>SC3</i> and ITC-2007 (Track 2) Benchmark Datasets	103
Figure 4.10 VND with a Perturbation	104
Figure 4.11 Perturbation Procedure for the Proposed VND Algorithm	105
Figure 4.12 Stage 2 Tabu Search Algorithm	108
Figure 5.1 Comparison Between the Number of Soft Constraint Violations that occurred using VND_Literature and VND_Proposed for Semester 1's Instance. $N = 30$ runs.	115

Figure 5.2	Comparison Between the Number of Soft Constraint Violations that occurred using VND_Literature and the VND_Proposed for Semester 2's Instance. $N = 30$ runs.	116
Figure 5.3	Comparison of the Number of Soft Constraint Violations that Occurred Using VND_Literature, VND_1 2 3 5 6 7 and VND_1 2 4 5 6 8 for Semester 1's Instance. $N = 30$ runs.	118
Figure 5.4	Comparison of the Number of Soft Constraint Violations that Occurred Using VND_Literature, VND_1 2 3 5 6 7 and VND_1 2 4 5 6 8 for Semester 2's Instance. $N = 30$ runs.	119
Figure 5.5	Comparison of the Soft Constraint Violations that Occurred Using VND_Proposed and VND_Proposed_P for Semester 1's Instance. $N = 30$ runs.	122
Figure 5.6	Comparison of the Soft Constraint Violations that Occurred Using VND_Proposed and VND_Proposed_P for Semester 2's Instance. $N = 30$ runs.	123
Figure 5.7	Original Timeslots Availability and Timeslots Availability After Three Hours Timeslots are Added	134
Figure 5.8	Original Timeslots Availability and Timeslots Availability After 3 Timeslots are Added Using Reserved Timeslots	136
Figure 5.9	Comparison Between the Number of Soft Constraint Violations that occurred using VND_Proposed and Tabu Search for Semester 1's Instance. $N = 30$ runs.	140
Figure 5.10	Comparison Between the Number of Soft Constraint Violations that occurred using VND_Proposed and Tabu Search for Semester 2's Instance. $N = 30$ runs.	141

LIST OF ABBREVIATIONS

UNIMAS	Universiti Malaysia Sarawak
FCSIT	Faculty of Computer Science and Information Technology

CHAPTER 1

INTRODUCTION

1.1 Study Background

Educational timetabling is defined as the task of allocating events such as exams, subjects and courses to rooms and timeslots by fulfilling certain constraints (Tan et al., 2021; Thepphakorn & Pongcharoen, 2020; Tan et al., 2020; Assi et al., 2018). Timetabling is a challenging combinatorial optimisation problem in theory and practice (Schaerf, 1999). Universiti Malaysia Sarawak (UNIMAS) devotes significant resources to the development of a feasible and high-quality course schedule for each faculty. Efficient course allocation may result in the more effective use of valuable resources such as utility resources (Burke et al., 2005). For example, reducing the number of timeslots needed to generate a feasible solution. This saves valuable resources, such as electricity costs when lectures can be avoided to be conducted at night time. Therefore, it is crucial to find an optimal configuration for the variables defined to achieve specific objectives (Habashi et al., 2018).

The university course timetabling problem (UCTTP) involves allocating a set of courses to limited resources - namely lecturers, venues and timeslots - by fulfilling certain constraints (Goh et al., 2020; Goh et al., 2019; Erdeniz & Felfernig, 2018; Goh et al., 2017). The UCTTP can be divided into two different categories based on the problem settings and requirements, namely the curriculum-based course timetabling problem (CBCTTP) and the post-enrolment course timetabling problem (PECTTP). The UCTTP at UNIMAS is closely related to the CBCTTP. Constraints can be classified into two types, hard and soft. The fulfilment of hard constraints is mandatory in generating a feasible timetable. For example, a student cannot attend two lectures at the same time, while a lecturer cannot lecture on more

than one course simultaneously. Meanwhile, the fulfilment of soft constraints is optional but will determine the quality of the timetable generated. For example, students should not have only one lecture hour per day and lecturers should not have to lecture after five pm.

The UCTTP is known to be NP-hard (Wang et al., 2019; Song et al., 2018; Assi et al., 2018; Gunawan, Ng, & Poh, 2012); that is, the problem cannot be solved exactly in polynomial time as the growth of the problem size and its complexity are exponential (Babaei et al., 2015; Bardadym, 1996). Exact algorithms are guaranteed to provide optimal solutions but they are applicable to small-sized problems (Schaerf, 1999). As an alternative, heuristic algorithms are often utilised to provide relatively good solutions in an acceptable time (Gora et al., 2010).

To date, many studies have been conducted on the UCTTP, tackling either benchmark or real-world UCTTPs. For most real-world search problems, automatically generating high-quality solutions is a difficult challenge (Muklason et al., 2019). Benchmark UCTTP are usually oversimplified and meant for objective comparison of methodologies. Meanwhile, real-world UCTTP focus on the practicality of solution at academic institutions. Even real-world UCTTP vary between them in terms of requirements due to different policies, education systems and cultures. The objective is to find a feasible timetable with the fewest possible soft constraint violations. Furthermore, the UCTTP requirements differ across academic institutions as policies and regulations are unique to each institution. This study addresses the UCTTP at the Faculty of Computer Science and Information Technology (FCSIT), UNIMAS, using a real-world dataset. The performance of the improved variable neighbourhood descent (VND) is investigated. Its performance is compared against the existing timetable, which had been constructed using commercial timetabling software.

1.2 Problem Statement

This research focuses on a real-world course timetabling problem at Universiti Malaysia Sarawak. The issues addressed are as follows:

i. Pre-registration data practice

At UNIMAS, there is no practice of course pre-registration. Student course registration only takes place after the timetable has been generated. Therefore, the timetable generated cannot cater to all the students individually.

Student enrolment data is predicted by the program coordinators by gut feeling. There are uncertainties in preparing the student-course registration data. Eventually, the venue utilisation is low, especially since a large-capacity venue may be allocated but a low number of students register. Other than that, “critical courses” involving “graduating students” cannot be identified, creating drastic changes to the planned timetable. Critical courses refer to courses that must be registered by repeating students for them to graduate.

ii. Curriculum-based course timetabling

Curriculum-based course timetabling means that the timetable generated is based on the curriculum plan for each program. This causes the “repeaters”, those who need to repeat the course/s they failed in the previous semester, to face a certain dilemma as they cannot attend both the repeated and current semester courses as these clash. The current timetable used by the FCSIT cannot cater to all the “repeater” students because each “repeater” is a unique case. For example, one student may need to repeat three courses from the previous semester, whereas another may only need to repeat one. In the worst case, a student may need to repeat courses from two or three previous semesters. The consequence may be that

the “repeaters” extend the duration of their study and are unable to graduate on time. This would also be a waste of time and money. On the other hand, the current timetable has also become highly rigid because of “repeaters”. For example, the timetable for the entire cohort of an intake may need to be adjusted because of one “repeater”.

iii. Size of the problem instance

The problem will increase in complexity when the number of intakes at UNIMAS increases. This is because when the number of students increases, a feasible schedule will be needed to satisfy more constraints. This causes more timeslots to be needed to generate a feasible schedule. The solution generated by existing timetabling software required 48 timeslots (8.00am – 9.00pm) to achieve feasibility. This consumes extra resources, such as electricity costs.

The number of students applying to FCSIT has skyrocketed. Sharing venues for lectures is a common feature of most academic institutions, especially when venues can accommodate a large number of students. This will cause a lack of venues with a high seating capacity. With each semester's increasing student enrolment, this situation will worsen. For example, the number of students enrolled at FCSIT for Semester 1 2019/2020 is 1454 whereas for Semester 1 2018/2019 is 1265. It is an increment of 15%. However, the number of venues remains the same.

1.3 Objectives

The objectives of this research are as follows:

- i. To formulate a simulation model of a real-world university course timetabling problem.

- ii. To improve variable neighbourhood descent algorithm to solve the real-world university course timetabling problem.
- iii. To conduct a sensitivity analysis as part of a comparison study of the proposed algorithm using varieties of the real-world university course timetabling problem.

1.4 Research Scope

The research scope is detailed from three different perspectives, namely the data, the algorithm and the solution evaluation.

The first perspective involves the data. The timetabling problem in this study is based on the real-world scenario at the Faculty of Computer Science and Information Technology (FCSIT), Universiti Malaysia Sarawak (UNIMAS). The courses offered by FCSIT can be divided into various categories, namely lectures, lectures with tutorials and lectures with lab sessions. These courses range from two to four credit hours. This study focuses on the lecture content of the faculty courses. Lab and tutorial sessions for the faculty courses were excluded in this study. Two real-world datasets were obtained from the faculty's administrator/ timetable planner. To demonstrate further the efficiency of the proposed algorithm, the International Timetabling Competition (ITC-2007-Track 2) benchmark dataset was utilised as a testbed. This benchmark dataset was chosen because its features are the closest in similarity to the real timetabling problem used in this study.

The second perspective is the algorithm. The UCTTP is known as an NP-hard problem, denoting that the problem cannot be solved exactly in polynomial time as the growth of the problem size and its complexity are exponential. Exact algorithms are

guaranteed to provide optimal solutions, but they apply to small-sized problems. This led to the utilisation of two-stage heuristic algorithms in this study as this would provide relatively good solutions in an acceptable time. In stage 1, heuristic orderings were utilised to find a feasible solution. In stage 2, the variable neighbourhood descent approach was utilised to improve the quality of the solution.

The last perspective is the solution evaluation. The timetable generated using the proposed algorithm was compared with the existing timetable. The International Timetabling Competition (ITC-2007-Track 2) dataset was used to demonstrate the significant improvement of the proposed algorithm. Sensitivity analysis was also performed on the heuristic proposed in this study. The robustness of the algorithm was analysed in terms of determining its feasibility after changing problem instance features such as the number of students, number of events, number of timeslots available for allocation, and number of large shared venues. The timetable generated using the proposed algorithm was also compared with the timetable generated using the Tabu Search (TS) algorithm.

1.5 Research Methodology

The flows of the research methodology used in this study are as follows:

Step 1: Dataset collection and interview

Interview session was conducted with the FCSIT administrator/timetable planner to collect the real-world dataset. All the details about the FCSIT timetabling problem were gathered, namely hard and soft constraints.

Step 2: Problem formulation

In this step, the sets and parameters were defined based on the FCSIT timetabling problem. Using set notations and parameter notations, the mathematical models for all the hard and soft constraints derived are illustrated. Next, an objective function was presented to calculate the number of soft constraint violations.

Step 3: Algorithm development

Step 3 involved algorithm development, where a computational bounded heuristic algorithm was developed based on the mathematical models illustrated in step 2. The algorithm was coded using visual basic (VB.Net). Microsoft Access was used as the database management software.

Step 4: Testing

After developing the algorithm, experiments were carried out to test its validity. Two types of datasets were used. The first was the real-world dataset obtained from the FCSIT administrator/timetable planner. The second dataset used was the ITC-2007 (Track 2) benchmark dataset.

Step 5: Conduct analysis

After checking the validity of the algorithm, computational analysis was undertaken to evaluate the performance of the proposed algorithm. Sensitivity analysis was also carried out on different variants of the problem to test the flexibility of the algorithm.

Step 6: Result Compilation

In the last step, a thesis with a full report was written, which includes the problem description, the mathematical model, the methodology, the computational bounded heuristic

algorithm that was developed, the computational results and the analysis of the proposed algorithm.

1.6 Research Significance

Through this research, it is hoped that a suitable algorithm can be formulated to solve the timetabling problem at UNIMAS. The goal is to achieve optimisation, whereby the resources can be used effectively. If the venue and timeslot allocation can be done effectively, the duration of the day in terms of lecture time can be reduced.

Meanwhile, “repeaters” must register for both courses that they need to repeat and their current courses with no timetable clashes. Without a good timetabling algorithm, the “repeaters” might need to drop a lot of their current courses due to timetable clashes. As a consequence, “repeaters” might need to extend the duration of their study and be unable to graduate on time. Other consequences are that the financial cost and duration of study are increased.

This research, which devised a good timetabling algorithm, is also needed to ensure students can concentrate on their studies. Long hours of studying with no breaks make students very tired. Even a lecturer will be unable to handle continuous lectures without a break. Therefore, it is important to have a distributed timetable.

1.7 Thesis Outline

This thesis consists of seven chapters. The content of the chapters is outlined as follows:

Chapter 2: Literature Review

Chapter 2 highlights the approaches that have been proposed by researchers to address the university course timetabling problem (benchmark and real-world). The approaches that have developed quality solutions to the timetabling problem are summarised in this chapter to provide better knowledge and understanding of these approaches.

Chapter 3: Case Study

A case study involving the Faculty of Computer Science and Information Technology is explored in this chapter. Problems identified in the current FCSIT timetabling method are discussed in detail. Based on the hard and soft constraints specified, the mathematical modelling for this study is illustrated. The general framework for solving the UCTTP at the FCSIT, UNIMAS, which consists of two stages, is displayed in a flowchart.

Chapter 4: Proposed Algorithm: Heuristic Orderings and Variable Neighbourhood Descent

Chapter 4 illustrates the algorithm used to find the feasibility in stage 1 (Heuristic Orderings) and the algorithm used to improve the quality of the solution in stage 2 (Variable Neighbourhood Descent). A detailed explanation of the algorithm is provided.

Chapter 5: Real-World University Course Timetabling Problem (FCSIT, Unimas)

This chapter presents the experimental results using a simulator that was developed based on the real-world scenario at the Faculty of Computer Science and Information Technology (FCSIT), Universiti Malaysia Sarawak (UNIMAS). The timetable generated using the proposed algorithm is then compared with the existing timetable. This chapter also discusses the experimental results and illustrates how the proposed algorithm performed with different varieties of the problem. The analysis of the final solution generated after changing

problem instances features is presented, such as the number of students, number of events, number of timeslots available for allocation and number of large venues.

Chapter 6: ITC-2007 (Track 2) Benchmark Dataset

This chapter presents the experimental results using a simulator developed based on the ITC-2007 (Track 2) benchmark dataset. This chapter also provides a discussion of the experimental results. A comparison between the proposed algorithm and Tabu Search was also presented.

Chapter 7: Conclusion

The last chapter concludes the overall research and includes future works that could be undertaken to enhance the solution found in this study. The contributions made by this study are explained in detail to highlight the importance of the current research.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This chapter highlights the most recent approaches in addressing the UCTTP. The mechanism of the approaches are briefly discussed and the achievement of the approaches are compared. The approaches are then categorised into Operational Research (OR) based techniques, meta-heuristics (single-solution and population-based approaches), hyper-heuristic approaches, multi criteria/ objective and hybrid approaches. We discuss the advantages and disadvantages of each category. We present the approaches in chronological order to show the trend in the UCTTP. In addition, the methodologies are grouped according to benchmark datasets to identify not only the popular datasets but the state-of-the-art methodologies for each dataset. Furthermore, we present case studies of real-world UCTTP. Constraints of different institutions are presented. Real-world UCTTP instances are unique due to different policies set by the institutions. The work in this chapter has been published as:

- Chen, M. C., Sze, S. N., Goh, S. L., Sabar, N. R., & Kendall, G. (2021). A Survey of University Course Timetabling Problem: Perspectives, Trends and Opportunities. *IEEE Access*, 9, 106515–106529. <https://doi.org/10.1109/ACCESS.2021.3100613>

2.2 Approaches/ Methodologies in Benchmark UCTTP

The methodologies utilised in UCTTP can be divided into six categories. The first category is operational research (OR) based techniques (graph colouring heuristics,

integer/linear programming, mixed integer linear programming and constraint logic programming). The second category is single solution-based meta-heuristics (tabu search, variable neighbourhood search and simulated annealing). The third category is population-based meta-heuristics (genetic algorithms, ant colony optimisation and particle swarm optimisation). The fourth, fifth and sixth categories are hyper-heuristic, multi criteria/objective and hybrid approaches.

2.2.1 Operational Research (OR) Based Techniques

Operational Research (OR) based techniques employ mathematical and analytical techniques in solving a challenging combinatorial optimisation problem. The graph colouring problem requires allocating minimal colours to vertices such that vertices connected by edges are allocated different colours. Timetabling and graph colouring are related such that events represent vertices, clashes between events correspond to edges and time slots denote colours (de Werra, 1985). All individual courses are referred to as events. Due to the interconnection of graph colouring problem and UCTTP, earlier algorithms were derived from graph colouring heuristics (Lewis, 2008; Burke, 1997). Lectures are assigned to periods (days and time) sequentially based on graph colouring heuristics for instance; largest degree, saturation degree, largest weighted degree and colour degree (Burke & Petrovic, 2002).

Conforming to the largest degree heuristic, events with the largest count of conflicts with other events should be assigned a period first as it is difficult to find a valid period for an event that has many clashes with other events. Based on the largest weighted degree heuristic, conflicting events with the highest number of students are assigned to the period first. In the saturation degree heuristic, the next event to schedule is the one with the lowest

remaining suitable periods. In colour degree heuristic, priority is given to events with the highest number of conflicts with the scheduled events. Both the saturation degree and colour degree heuristics are calculated dynamically.

Liu et al. (2011) proposed a clique-based algorithm to generate feasible solutions for UCTTP. The clique refers to a set of courses that could be allocated in the same timeslot. Recombination and perturbation steps were taken to increase the size of the clique generated. The proposed algorithm was tested using hard benchmark datasets. The algorithm was comparable with other effective algorithms.

Budiono and Wong (2012) proposed a Graph Colouring (GC) approach to find feasible solutions for UCTTP. The proposed algorithm had two stages. In stage one, Least Saturation Degree First (LSDF) was used in finding feasible solutions. In stage two, the solution quality was improved using operators based on a column permutation. The algorithm was tested using Socha benchmark datasets. The algorithm managed to produce encouraging results.

Lindahl et al. (2018) proposed a Mixed Integer Linear Programming (MILP) approach to solve the CB-CTTP. Three bi-objective mixed-integer models were formulated. Problem instances from ITC-07 (Track 3) were used as testbeds. They found that the objectives (rooms, teaching periods and solution quality) affect one another and the relationships between these objectives are dependent on the problem instances.

Bagger et al. (2019) proposed an Integer Programming (IP) relaxation to solve CB-CTTP. The model formulated was called pattern formulation, where a course was assigned to a set of periods on one day. The proposed model was tested using ITC-07 (Track 3)

benchmark datasets. The proposed model managed to improve the lower bounds for three of the problem instances.

2.2.2 Single Solution-based Meta-heuristics

Single solution-based meta-heuristics are known as problem-independent high-level algorithmic framework that gives a set of tactics for constructing heuristic optimisation algorithms (Sørensen and Glover, 2013). Single solution-based meta-heuristics operate on a single solution and apply various operators and rules to evolve a new solution in the neighbourhood areas of current one. Local search algorithms start the search with a single solution and then explore its neighbourhood areas to find a better one. Examples of local search algorithms are tabu search, simulated annealing, hill climbing and iterated local search. Lewis (2008) categorised single solution-based meta-heuristics into three types of optimisation algorithms which are one-stage, two-stage and the one that allows relaxation. One-stage optimisation algorithms satisfy both hard and soft constraints at the same time. The best solution is determined by using a weighted sum function where each constraint is given a penalty value based on its importance. In two-stage optimisation algorithms, only hard constraints are considered in obtaining a feasible solution in the first stage. Meanwhile, only soft constraints are considered in getting a high quality solution in the second stage. For algorithms that allow relaxation, hard and soft constraint violations are addressed by relaxing some aspects of the problem instance. There are two types of relaxation. The first temporarily put aside events that cannot be scheduled in a feasible solution. The second creates dummy or extra time slots to artificially accommodate the events to create a feasible solution (Kiefer et al., 2017).

(a) Tabu Search

Tabu search (TS) uses a tabu list to avoid being stuck in local optima. Whenever it is trapped in local optima, the search continues with non-improving moves but solutions generated before will be rejected with the use of a tabu list.

Nagata (2018) proposed a TS algorithm with the ability to change the neighbourhood size called Random Partial Neighbourhood Search (RPNS) to solve the PE-CTTP. The proposed algorithm was tested using Socha, ITC-02 and ITC-07 (Track 2) benchmark datasets. The algorithm produced competitive results when compared to leading solvers.

(b) Simulated Annealing

Simulated Annealing (SA) accepts non-improving moves using probabilistic acceptance criteria. Its performance is dependent on the initial and end temperatures, cooling schedule and definition of neighbourhood structures.

Lewis (2010) proposed a time-dependent SA algorithm to solve the PE-CTTP. The algorithm had three distinguishable stages. Each stage had its defined time limit and the total running time must not exceed the full time limit. If one stage is completed earlier than the specified time limit, the extra time could be utilised in the next stage. At each stage, constraints satisfied in the previous stages must not be violated. Stage 2 and stage 3 were implemented using SA. The proposed algorithm was tested using ITC-07 (Track 2) benchmark datasets.

Ceschia et al. (2012) proposed a SA algorithm to solve the PE-CTTP. Two neighbourhood structures were used moving one event and swapping two events. Moving one event means reassigning the event to another feasible timeslot. The proposed algorithm

was tested using Socha, ITC-02, ITC-07 (Track 2) and hard benchmark datasets. A well engineered and finely tuned solver managed to outperform most of the algorithms in the scientific literature.

Goh et al. (2017) proposed Tabu Search with Sampling (TSSP) and Simulated Annealing with Reheating (SAR) to address the PE-CTTP. In stage one, TSSP was utilised to create feasible solutions. In stage two, SAR was used to improve the quality of the solutions. The algorithm was tested with Socha, ITC-02 and ITC-07 (Track 2) benchmark datasets. It managed to produce the new best solutions for many instances.

Goh et al. (2019) proposed SA with Improved Reheating and Learning (SAIRL) in addressing the PE-CTTP. The method consisted of two stages. In the first stage, a feasible solution was generated, which was improved in the second stage. A reinforcement learning-based methodology was proposed to obtain a suitable composition of neighbourhood structures. The proposed algorithm was tested using Socha, ITC-02 and ITC-07 (Track 2) datasets. The approach managed to generate six new best results.

Goh et al. (2020) proposed a two-phase hybrid local search algorithm to solve PE-CTTP. In the first phase, TSPP and Iterated Local Search (ILS) were used to generate a feasible solution. In the second phase, SAR with two preliminary runs (SAR-2P) was used to improve the quality of the solution. Information gathered from the preliminary runs helped to improve the efficiency of SAR. The algorithm was tested with Hard, Socha, ITC-02 and ITC-07 (Track 2) benchmark datasets. It produced three new best results and seven new mean results.

(c) Iterated Local Search

Song et al. (2018) proposed an Iterated Local Search (ILS) algorithm to address the UCTTP. It consisted of three stages which were initialisation, intensification and diversification. The 60 instances from Lewis and Paechter (2007) were used. The proposed algorithm managed to find feasible solutions for 58 instances.

2.2.3 Population-based Meta-heuristics

Population-based meta-heuristics operate on a population of solutions and apply various operators and rules to evolve a new population of solutions in the neighbourhood areas of the current ones. Examples of population-based meta-heuristics are genetic algorithms, ant colony optimisation and particle swarm optimisation.

(a) Genetic Algorithms

Genetic Algorithm (GA) tackles optimisation problems using the concept of biological evolution. In each iteration, the algorithm selects solutions from the population using selective pressure to bias it towards choosing the best members as parents and these parent solutions are used to generate children solutions for the next generation (Harada and Alba, 2020). This process is iterated until an optimal solution is generated or the time allowed has expired. In each iteration, the algorithm conforms to selection rules, crossover rules and mutation rules. Selection rules manage the selection of parents from the current population. Crossover rules are ways that parents are combined in generating children solutions for the next generation. Mutation is the way solutions are randomly changed to motivate diversity in the population.

Assi et al. (2018) proposed a GA approach in tackling the UCTTP. An initial population was generated by randomly assigning classes to periods by considering room capacity. The fitness value of the parents would determine their selection in generating children for the next generation through crossover and mutation operators. The authors used an array of classes in representing the chromosome where information such as the lecturers, rooms and periods were stored. This representation avoided conflict between courses. The approach was tested using their own generated data. Solutions were improved iteratively.

(b) Ant Colony Optimisation

Ant Colony Optimisation (ACO) is an approach inspired by observing the foraging behaviour of ants (Dorigo and Stutzle, 2009). To mark the shortest path in finding or transporting food, the ants deposit pheromone along the path as a guideline for other ants to follow. In optimisation problems, the artificial ants build their solutions and share the information on the quality of their solution with other artificial ants.

Nothegger et al. (2012) proposed an ACO approach to tackle the PE-CTTP. Pheromone information was stored in two distinct matrices. Events were chosen randomly and allocated to time slots and rooms according to pheromone information. The solution was then further improved by an ejection chain. The pheromone information was updated accordingly based on the solutions with promising soft constraints penalty (SCP) and distance to feasibility (DTF) scores. The algorithm was tested using ITC-07 (Track 2) benchmark datasets. It outperformed many algorithms such as Simulated Annealing (SA) and Tabu Search (TS).

Badoni and Gupta (2016) proposed an ACO approach to address the UCTTP by grouping students in mutually exclusive groups and then assigning each group to timeslots

and venues accordingly. Three steps were iterated namely the development of initial solutions by the artificial ant, pheromone update and the execution of the local search. The proposed method was tested using Socha datasets. The computational times were acceptable compared to other existing algorithms such as Great Deluge (GD), Genetic Algorithm (GA), hyper-heuristic and hybrid.

(c) Particle Swarm Optimisation

Particle Swarm Optimisation (PSO) originated from social behaviour shown by collective species such as the flocking of birds, a group of tigers and a swarm of bees (Teoh et al., 2015). Every individual in a PSO model is called a particle and each particle position is equivalent to a candidate solution with a designated fitness function for the optimisation problem. This algorithm gives extra attention towards exploration and exploitation of the search space (Ghalia, 2008). The efficiency of PSO can be improved by controlling the parameters such as the size of the swarm, inertia weight and acceleration coefficients.

Chen and Shih (2013) proposed a constriction PSO to address the UCTTP. They worked on their own generated data. Among the advantages of PSO reported were fast convergence, fewer parameter settings and the ability to set dynamic environment characteristics. In generating a good quality solution, an interchange heuristic was applied to ensure a better exploration of the solution space. The interchange heuristic prevented the particles from being trapped in local optima and allowed faster convergence to global optima. The method managed to generate acceptable solutions.

(d) Fish Swarm Intelligent

Turabieh et al. (2010) proposed fish swarm intelligent to solve the UCTTP. The proposed algorithm simulated movement shown by fish when searching for food. The search space was categorised into crowded, not crowded and empty areas. Each solution in the solution population was represented by a fish. Two local searches were used to improve the quality of the solution namely multi decay rate great deluge and steepest descent. The proposed algorithm was tested using Socha benchmark datasets. The algorithm produced best known results for some of the instances.

(e) Honey-bee Mating

Sabar et al. (2012) proposed a honey-bee mating algorithm to solve the PE-CTTP. The proposed algorithm simulated the behaviour shown by honey-bees when mating. This algorithm is effective in exploring and exploiting the search space at the same time. The proposed algorithm was tested using Socha benchmark datasets. Best known results were reported for some of the instances.

(f) Population Based Local Search

Abuhamdah et al. (2013) proposed Population Based Local Search (PB-LS) to solve the UCTTP. They claimed it was good in exploring and exploiting the search space. Two operators were utilised for searching, namely single-direction and all-direction force. The proposed algorithm was tested using Socha benchmark datasets. The algorithm outperformed other approaches such as Simulated Annealing (SA), Tabu Search (TS), Great Deluge (GD), Genetic Algorithm (GA), hyper-heuristic, hybrid, Fish Swarm Intelligent and Variable Neighbourhood Search (VNS).

2.2.4 Hyper-heuristic

Hyper-Heuristic approaches employ several meta-heuristics in adaptive manner to solve the problem at hand (Sabar et al., 2020; Turkey et al., 2020). Habashi et al. (2018) proposed Add-Delete Hyper-Heuristic (ADHH) to solve the UCTTP. The approach used an adaptive heuristic generation method through a variable-sized list of add and delete operations. The approach was tested on ITC-07 benchmark datasets. Its performance was better, on average, compared to other algorithms in the scientific literature.

2.2.5 Multi Criteria/ Objective Approaches

Multi-Criteria/ Objective approaches are used to solve combinatorial optimisation problems with multiple conflicting criteria or objectives. Gülcü and Akkan (2020) proposed a Multi-Objective Simulated Annealing (MOSA) in tackling the UCTTP. They aimed to define a good Pareto front by taking into consideration the solution quality and the robustness of the solution. Algorithms were developed with single and multiple disruptions. Single disruption referred to disruption of only one lecture whereas multiple disruptions referred to disruption of more than one lecture. The ITC-07 (Track 3) benchmark dataset was used as a testbed. The algorithm with multiple disruptions outperformed the one with a single disruption.

2.2.6 Hybrid Approaches

Hybrid approaches combine the effective characteristics of different approaches to solve combinatorial optimisation problems. Shaker and Abdullah (2010) proposed a Round Robin Scheduling Algorithm (RR) to control SA, Great Deluge (GD) and Hill Climbing (HC) in solving the UCTTP. It improved the quality of the initial solution generated using

the least saturation degree heuristic. The algorithm was tested on Socha benchmark datasets. It managed to produce competitive results in comparison to other state-of-the-art methods.

Lü and Hao (2010) proposed an Adaptive Tabu Search (ATS) in addressing the CB-CTTP. The framework consisted of three stages. In the first initialisation stage, an initial feasible solution was generated using a greedy algorithm. A tabu search algorithm was used as a search intensification in the second stage. In the third stage, a perturbation operator from an iterated local search (ILS) was used as a diversification mechanism. Both intensification and diversification were employed to minimise soft constraint violations. They worked on ITC-07 (Track 3) datasets. The proposed method managed to produce better results compared to the TS and ILS run individually.

Akkan and Gülcü (2018) proposed a Hybrid Genetic Algorithm (HGA) in addressing the CB-CTTP. Hill climbing, simulated annealing and a genetic algorithm were hybridised in solving the formulated problem. Three moves were implemented during disruption; simple move, swap move and Kempe chain move. They tested the hybrid algorithm on ITC-07 benchmark datasets. The results produced showed high-quality Pareto fronts.

Kampke et al. (2019) proposed a network flow technique for the UCTTP. They generated a local solution by using Greedy Randomised Adaptive Search Procedure (GRASP) constructive algorithm together with a maximum flow partial solution. In their work, CB-CTTP was re-modelled using the maximum network flow technique. The local solution was then improved in terms of quality by using simulated annealing. The proposed method generated competitive solutions for the ITC-07 (Track 3) instances.

2.3 Approaches/ Methodologies in Real-world UCTTP

2.3.1 Operational Research (OR) Based Techniques

Prabodanie (2017) proposed an Integer Programming (IP) model for the UCTTP at the Faculty of Applied Sciences, Wayamba University of Sri Lanka. The primary objective was to minimise the number of working hours per week. Decision variables were defined using a relational matrix. The model included completeness, uniqueness, consecutive and pre-assignment constraints. OpenSolver and Microsoft Excel were used as simulators. Results showed that UCTTP could be formulated effectively using fewer decision variables and constraints. The model managed to reduce the number of working hours per week. The quality of the timetable could be further improved by considering the preferences of students and teachers.

Avella et al. (2019) proposed a Mixed Integer Programming (MIP) to solve the CB-CTTP at the Engineering Department of Sannio University. Two local branching heuristics were used. The first heuristic changed the room allocation but not the time-slot allocation. The second heuristic changed the day allocation but not the room and time-slot allocation. The proposed algorithm was tested on data derived from two semesters. It was shown to be efficient.

2.3.2 Single Solution-based Meta-heuristics

(a) Tabu Search

Aladag et al. (2009) proposed a Tabu Search (TS) algorithm in solving UCTTP for the Department of Statistics at Hacettepe University by utilising four neighbourhood structures. They were simple move, swap move and combination of both moves called Mixed_1 and Mixed_2. From experiments, simple move and Mixed_1 managed to generate

the best timetables. Figure 2.1 shows the simple move and the swap move proposed by Aladag et al. (2009).

Day\Time	8-9	9-10	10-11	11-12	12-13	13-14	14-15	15-16	16-17
Mon	1	2	3	4	5	6	7	8	9
Tue	10 LectA	11 LectA	12 LectA	13	14	15	16	17	18
Wed	19	20	21	22	23	24	25	26	27
Thurs	28	29	30	31	32	33	34	35	36
Fri	37	38	39	40	41	42 LectA	43 LectA	44 LectA	45

(a) The Simple Move

Day\Time	8-9	9-10	10-11	11-12	12-13	13-14	14-15	15-16	16-17
Mon	1	2	3	4	5	6	7	8	9
Tue	10 LectA	11 LectA	12 LectA	13	14	15	16	17	18
Wed	19	20	21	22	23	24 LectB	25 LectB	26 LectB	27
Thurs	28	29	30	31	32	33	34	35	36
Fri	37	38	39	40	41	42	43	44	45

(b) The Swap Move

Figure 2.1: Neighbourhood Structures

(b) Simulated Annealing

Aycan and Ayav (2009) proposed a Simulated Annealing (SA) method in finding a feasible timetable for the Department of Computer Engineering in Izmir Institute of Technology. The authors investigated the performance of neighbourhood searching algorithms called swapping, simple search and their combinations. The performance of these algorithms was compared in terms of computational times and total costs. The datasets used were taken from 2007 to 2008. Results showed that the combination of a simple search, swapping and simple search-swapping produced the most satisfactory timetable.

(c) Variable Neighbourhood Descent

Borchani et al. (2017) proposed a Variable Neighbourhood Descent (VND) approach in addressing the UCTTP for the Faculty of Economics and Management Sciences of Sfax in Tunisia. The objectives were to minimise the total number of holes and the number of isolated lessons. Eleven neighbourhood structures were developed. Six neighbourhood structures to solve holes and five neighbourhood structures to solve isolated lessons. Six real datasets from 2012 to 2014 were used. Results showed that on average, the approach managed to eliminate 52.47% of the holes and isolated lessons. The quality of the feasible solution could be further improved by minimising working days and allocating lunch breaks for students.

2.3.3 Population-based Meta-heuristics

(a) Genetic Algorithms

Almeida et al. (2016) proposed a Genetic Algorithm (GA) approach to improve the quality of timetable for the Information Systems program of Federal University of Rio Grande do Norte (UFRN). Among the requirements were minimising time gaps between non-consecutive lectures for a group of students and avoiding scheduling all lectures for a course on the same day. Real datasets from the first semester of 2012 to the second semester of 2015 were used. The performance of the algorithm was determined by its fitness function. Results showed that the solutions generated were better or equal to the ones generated manually.

2.3.4 Hyper-heuristic

Muklason et al. (2019) proposed a Hyper-Heuristic (HH) algorithm in addressing the UCTTP for the Department of Information Systems, Institut Teknologi Sepuluh Nopember, Indonesia. The objective was to generate a highly flexible optimal solution. The hyper-heuristic approach combined a TS and a VNS. Two datasets from 2017/2018 were used. Results produced were superior to that of the manual timetable in terms of soft constraint violations.

2.3.5 Hybrid Approaches

Shatnawi et al. (2012) proposed a clustering and colour mapping approach in producing a timetable for the College of Applied Studies at the University of Bahrain. The objective was to enable students to register for their courses without any clashes. Data from the official university registration system was extracted and processed in generating the timetable. The proposed algorithm was tested on problem instances featuring a total of 1270 students, 8 academic programs and 83 courses. Clusters of students were generated as an initial solution using data mining techniques. Next, they obtained the solutions using colour mapping algorithm. This was an improvement to the previous work by Albalooshi and Shatnawi (2010) which comprised the data mining component only.

Sze et al. (2017) proposed a two-stage heuristic with clustering for Universiti Malaysia Sarawak (UNIMAS) CB-CTTP. The objectives were to automate course timetabling and increase venue utilisation. The algorithm included a prediction on course registration by students. In stage one, courses were divided into different groups. In stage two, courses in a group were assigned to the same timeslot but different venues. Real datasets

from three semesters were used. Valid solutions with minimum unallocated courses were generated.

Matias et al. (2019) proposed a Hybrid Genetic Algorithm (HGA) with four neighbourhood operators to tackle the UCTTP for higher education institutions in the Philippines. The algorithm helped in managing the teaching workload. When new staff were hired, it was necessary to make sure the new staff were assigned to classes that could utilise their skills and would not cause the timetable to be infeasible. The dataset consisted of 118 classes, 308 hours workload per week, 45 timeslots, five laboratories and five lecture rooms. The algorithm managed to generate feasible solutions and optimize teaching workloads.

Vianna et al. (2020) proposed a hybrid of Variable Neighbourhood Search (VNS) and Tabu Search (TS) to address the UCTTP for Federal Fluminense University. The proposed algorithm was developed using the FINESS framework. The datasets used were derived from two undergraduate courses. Results indicated that the hybrid was better than the VNS and TS run individually.

Figure 2.2 shows the case studies of the real-world UCTTP. Constraints (hard and soft) are highly variable according to institutions. The same constraint may be hard/soft for one institution but soft/hard for the others. For example, the constraint “student should not have class for the last timeslot of the day” maybe considered hard constraint for Institution A but for Institution B, the constraint maybe considered as soft constraint. The type of constraints (soft/hard) are distinct among institutions where its depend on the policy or priority of the institution.

Author	Institution	Method	Constraints (HC - hard constraint; SC - soft constraint)													
			Lectures of all courses assigned to distinct periods	Room occupancy	Conflict in curricula & lecturer	Lecturer availability	Room capacity	Minimum working days	Curriculum compactness	Room stability	Preassigned periods	Lecturer preferences	Room optimization	No lecture at the last timeslots	Venue feature and availability	Student preferences
Aycan and Ayav (2009)	Department of Computer Engineering in Izmir Institute of Technology	SA	HC	HC	HC	HC			HC		HC	SC			SC	
Aladag et al. (2009)	Department of Statistics at Hacettepe University	TS	HC	HC	HC		SC	SC	SC		HC	SC		HC	SC	
Shanawi et al. (2012)	College of Applied Studies in University of Bahrain	Clustering and colour mapping	HC	HC	HC		HC	SC	SC				SC	HC		
Almeida et al. (2016)	Federal University of Rio Grande do Norte (UFRN)	GA	HC	HC	HC	HC		SC				SC			SC	
Prabodanie (2017)	Faculty of Applied Sciences, Wayamba University of Sri Lanka	IP	HC	HC	HC		HC	HC	HC		HC					
Sze et al. (2017)	Universiti Malaysia Sarawak (UNIMAS)	Clustering	HC	HC	HC		HC	SC	HC			SC			SC	SC
Borchani et al. (2017)	Faculty of Economics and Management Sciences of Sfax in Tunisia	VND	HC	HC	HC	HC	HC	SC	SC					HC		
Muklason et al. (2019)	Department of Information Systems, Institut Teknologi Sepuluh Nopember, Indonesia	HH (TS + VNS)	HC	HC	HC		HC	SC	SC				SC			
Matias et al. (2019)	Higher Education Institutions at Philippines	HGA	HC	HC	HC	SC	SC	SC					SC	SC	SC	
Avella et al. (2019)	Engineering Department of Sannio University	MIP	HC	HC	HC	HC		HC	HC	HC				SC	HC	
Vianna et al. (2020)	Federal Fluminense University	Hybrid (VNS + TS)	HC	HC	HC	HC		SC	SC					SC	SC	SC

Figure 2.1: Case Studies of the Real-world UCTTP

2.4 Perspectives in UCTTP

In this section, we provide some perspectives on UCTTP. Table 2.1 shows the approaches in addressing benchmark and real-world UCTTP. Figure 2.3 shows the classification of these approaches.

Table 2.1: Approaches in Solving UCTTP (Benchmark and Real-world Problem Datasets)

Year	Reference	Method	Category	Dataset
2010	(Lü & Hao, 2010)	ATS	Hybrid	ITC-07 (T3)
	(Turabieh et al., 2010)	Fish swarm intelligent	Population-based Meta-heuristic	Socha
	(Shaker & Abdullah, 2010)	RR scheduling algorithm (HC, GD, SA)	Hybrid	Socha
	(Lewis, 2010)	Time-dependent meta-heuristic	Single Solution-based Meta-heuristic	ITC-07 (T2)
2011	(Liu et al., 2011)	Clique-based	OR	Hard
2012	(Budiono & Wong, 2012)	GC	OR	Socha
	(Sabar et al., 2012)	Honey-bee mating	Population-based Meta-heuristic	Socha
	(Ceschia et al., 2012)	SA	Single Solution-based Meta-heuristic	Hard, Socha, ITC-02 and ITC-07 (T2)
	(Nothegger et al., 2012)	ACO	Population-based Meta-heuristic	ITC-07 (T2)
2013	(Chen & Shih, 2013)	PSO	Population-based Meta-heuristic	Own dataset
2014	(Abuhamdah et al., 2013)	PB-LS	Population-based Meta-heuristic	Socha
2016	(Badoni & Gupta, 2016)	ACO	Population-based Meta-heuristic	Socha
2017	(Goh et al., 2017a)	TSSP and SAR	Single Solution-based Meta-heuristic	Socha, ITC-02 and ITC-07 (T2)
2018	(Assi et al., 2018)	GA	Population-based Meta-heuristic	Own dataset
	(Song et al., 2018)	ILS	Single Solution-based Meta-heuristic	Hard
	(Akkan & Gülcü, 2018)	HGA, SA and HC	Hybrid	ITC-07 (T3)
	(Lindahl et al., 2018)	MILP	OR	ITC-07 (T3)
	(Habashi et al., 2018)	ADHH	Hyper-heuristic	ITC-07 (T3)
	(Nagata, 2018)	TS(RPNS)	Single Solution-based Meta-heuristic	Socha, ITC-02 and ITC-07 (T2)

Table 2.1 continued

2019	(Kampke et al., 2019)	Network flow, GRASP and SA	Hybrid	ITC-07 (T3)
	(Bagger et al., 2019)	ILP	OR	ITC-07 (T3)
	(Goh et al., 2019)	SAIRL	Single Solution-based Meta-heuristic	Socha, ITC-02 and ITC-07 (T2)
2020	(Gülcü & Akkan, 2020)	MOSA	Multi Criteria/Objective	ITC-07 (T3)
	(Goh et al., 2020)	SAR, ILS and SAR-2P	Single Solution-based Meta-heuristic	Hard, Socha, ITC-02 and ITC-07 (T2)
2009	(Aycan & Ayav, 2009)	SA	Single Solution-based Meta-heuristic	Real-world dataset
	(Aladag et al., 2009a)	TS	Single Solution-based Meta-heuristic	Real-world dataset
2012	(Shatnawi et al., 2012)	Clustering and color mapping	Hybrid	Real-world dataset
2016	(Almeida et al., 2016)	GA	Population-based Meta-heuristic	Real-world dataset
2017	(Prabodanie, 2017)	IP	OR	Real-world dataset
	(Sze et al., 2017)	Clustering	Hybrid	Real-world dataset
	(Borchani et al., 2017)	VND	Single Solution-based Meta-heuristic	Real-world dataset
2019	(Muklason et al., 2019)	HH (TS + VNS)	Hyper-heuristic	Real-world dataset
	(Matias et al., 2019)	HGA	Hybrid	Real-world dataset
	(Avella et al., 2019)	MIP	OR	Real-world dataset
2020	(Vianna et al., 2020)	VNS + TS	Hybrid	Real-world dataset

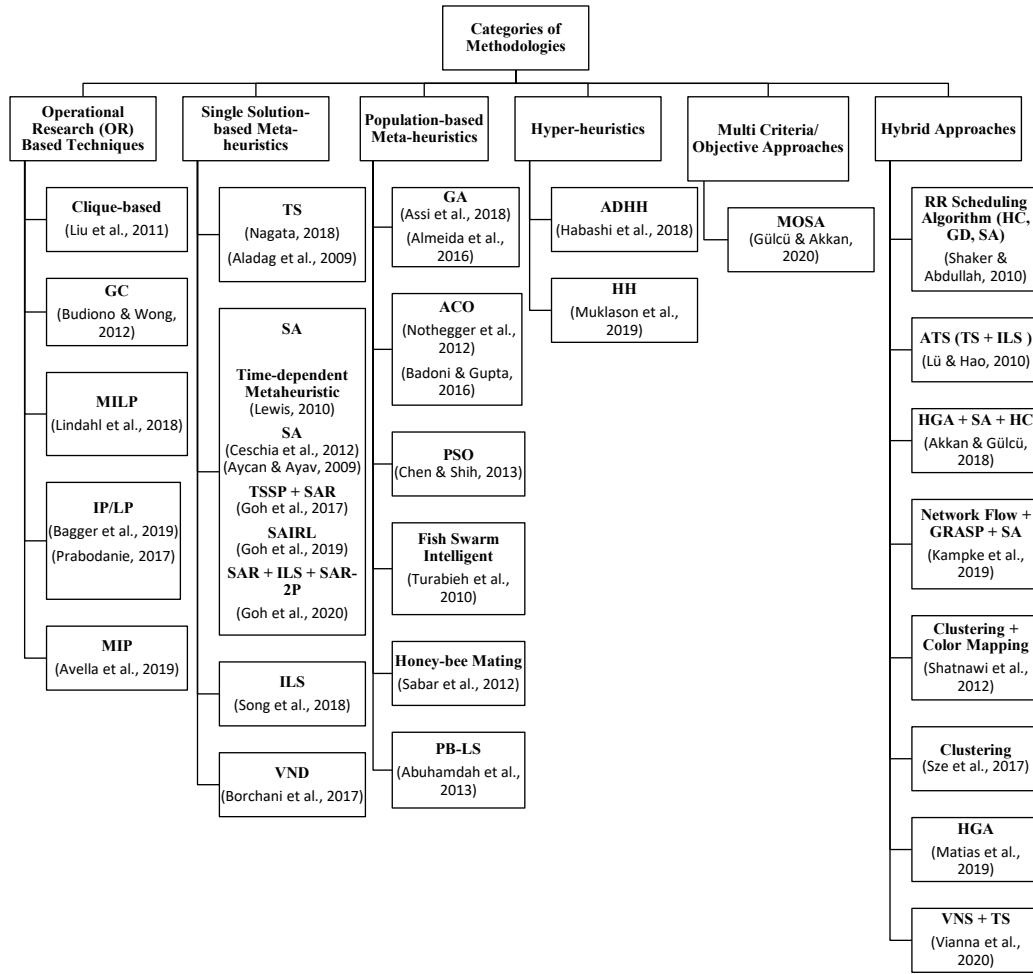


Figure 2.3: Classification of Approaches in Benchmark and Real-world UCTTP

As evident from Table 2.2, from the 35 papers surveyed, there are six OR methodologies, ten single solution-based meta-heuristics, eight population-based meta-heuristic approaches, two hyper-heuristics, one multi criteria/ objective and eight hybrid approaches.

For the benchmark UCTTP, single solution-based meta-heuristics(7) and population-based meta-heuristics(7) are the most popular approaches. Five out of the seven single solution-based meta-heuristic approaches are based on SA. Two out of the seven population-based meta-heuristic approaches are ACOs. It would be interesting to see the outcome of

hybridising SA with population-based approaches such as ACO which are popular for their explorative capability. Approaches such as hyper-heuristic and multi-criteria/objective are less popular perhaps due to their performance. However, they are less researched therefore providing the opportunity for new studies.

For the real-world UCTTP, hybrid(4) approaches are the most popular. This is followed by single solution-based meta-heuristic(3), OR (2), population-based meta-heuristic(1) and hyper-heuristic(1). Two out of the four hybrid approaches are hybrids of VNS and TS.

From observation, state-of-the-art approaches in benchmark UCTTP are not fully utilised in real-world UCTTP. Researchers may adopt/adapt the state-of-the-art approaches in benchmark UCTTP to real-world UCTTP at academic institutions.

Table 2.2: Summary of Approaches in UCTTP

UCTTP	Category						Total
	OR	Single Solution-based Meta-heuristic	Population-based Meta-heuristic	Hyper-heuristic	Multi Criteria/Objective	Hybrid	
Benchmark	4	7	7	1	1	4	24
Real-world	2	3	1	1	0	4	11
Total	6	10	8	2	1	8	35

2.5 Trends in Benchmark UCTTP

The benchmark datasets and their respective state-of-the-art methodologies are discussed in this section. As evident from Table 2.3, the benchmark datasets utilised in the international timetabling competitions are the most popular testbeds among researchers in comparing algorithms.

Table 2.3: Benchmark Datasets and Their Respective State-Of-The-Art Methodologies

Dataset	Year	Methods
Socha	2010	Fish Swarm Intelligent (Turabieh et al., 2010)
		RR scheduling algorithm (HC, GD, SA) (Shaker & Abdullah, 2010)
	2012	Honey-bee mating (Sabar et al., 2012)
		SA (Ceschia et al., 2012)
		GC(LSDF) (Budiono & Wong, 2012)
	2014	PB-LS (Abuhamdah et al., 2013)
	2016	ACO (Badoni & Gupta, 2016)
	2017	TSSP and SAR (Goh et al., 2017a)
	2018	TS(RPNS) (Nagata, 2018)
	2019	SAIRL (Goh et al., 2019a)
	2020	SAR, ILS and SAR-2P (Goh et al., 2020)
ITC-02	2012	SA (Ceschia, Gaspero, et al., 2012)
	2017	TSSP and SAR (Goh et al., 2017a)
	2018	TS(RPNS) (Nagata, 2018)
	2019	SAIRL (Goh et al., 2019a)
	2020	SAR, ILS and SAR-2P (Goh et al., 2020)
ITC-07 (Track 2)	2010	Time-dependent meta-heuristic (Rhyd Lewis, 2010)
	2012	SA (Ceschia, Gaspero, et al., 2012)
		ACO (Nothegger et al., 2012)
	2017	TSSP and SAR (Goh et al., 2017a)]
	2018	TS(RPNS) (Nagata, 2018)
	2019	SAIRL (Goh et al., 2019a)
	2020	SAR, ILS and SAR-2P (Goh et al., 2020)
ITC-07 (Track 3)	2010	ATS (Lü & Hao, 2010)
	2018	HGA, SA, HC (Akkan & Gülcü, 2018)
		MILP (Lindahl et al., 2018)
		ADHH (Habashi et al., 2018)
	2019	ILP (Bagger et al., 2019)
		Network flow, GRASP and SA (Kampke et al., 2019)
	2020	MOSA (Gülcü & Akkan, 2020)
Hard	2011	Clique-based (Liu et al., 2011)
	2012	SA (Ceschia, Gaspero, et al., 2012)
	2018	ILS (Song et al., 2018)
	2020	SAR, ILS and SAR-2P (Goh et al., 2020)

2.5.1 Socha Benchmark Dataset

The Socha benchmark dataset is developed by utilising an algorithm created by Ben Paechter. It consists of 11 instances. The features of this dataset are shown in Table 2.4. In the last 10 years, 11 different approaches were proposed for this dataset. Variants of SA proposed by Goh et al. (2020) are superior to others in terms of performance. Another state-of-the-art method for this dataset is the TS based approach called random partial neighbourhood search (RPNS) by Nagata (2018).

Table 2.4: The Features of the Socha Benchmark Dataset

Instances	Students	Events	Rooms	Features
S01	80	100	5	5
S02	80	100	5	5
S03	80	100	5	5
S04	80	100	5	5
S05	80	100	5	5
M01	200	400	10	5
M02	200	400	10	5
M03	200	400	10	5
M04	200	400	10	5
M05	200	400	10	5
L	400	400	10	10

2.5.2 ITC-02 Benchmark Dataset

The International Timetabling Competition 2002 (ITC-02) is organized by the Meta-heuristic Network and sponsored by the Practice and Theory of Automated Timetabling (PATAT). The benchmark dataset (20 instances) can be downloaded from the ITC-02 website with the URL <http://sferics.idsia.ch/Files/ttcomp2002/oldindex.html>. The dataset is produced using an algorithm by Ben Paechter. There is a time limit requirement for this dataset which is dictated by running a program on the host computer. Over the last 10 years, five different approaches have been proposed for this dataset. TSSP, ILS and SAR-2P

proposed by Goh et al. (2020) performed better than the other four. The features of this dataset are presented in Table 2.5.

Table 2.5: The Features of the ITC-02 Benchmark Dataset

Instances	Students	Events	Rooms	Features
01	200	400	10	10
02	200	400	10	10
03	200	400	10	10
04	300	400	10	5
05	300	350	10	10
06	300	350	10	5
07	350	350	10	5
08	250	400	10	5
09	220	440	11	6
10	200	400	10	5
11	220	400	10	6
12	200	400	10	5
13	250	400	10	6
14	350	350	10	5
15	300	350	10	10
16	220	440	11	6
17	300	350	10	10
18	200	400	10	10
19	300	400	10	5
20	300	350	10	5

2.5.3 ITC-07 (Track 2) Benchmark Dataset

The PE-CTT variant benchmark dataset (24 instances) from the International Timetabling Competition 2007 (ITC-07) can be downloaded from <http://www.cs.qub.ac.uk/itc2007/index.htm>. In the last 10 years, seven different approaches were proposed for this dataset. The current state-of-the-art methods are RPNS (Nagata, 2018), SAR-2P (Goh et al., 2020) and SA (Ceschia, Gaspero, et al., 2012). The features of the dataset are given in Table 2.6.

Table 2.6: The Features of the ITC-07 (Track 2) Benchmark Dataset

Instances	Students	Events	Rooms	Features
01	500	400	10	10
02	500	400	10	10
03	1000	200	20	10
04	1000	200	20	10
05	300	400	20	20
06	300	400	20	20
07	500	200	20	20
08	500	200	20	20
09	500	400	10	20
10	500	400	10	20
11	1000	200	10	10
12	1000	200	10	10
13	300	400	20	10
14	300	400	20	10
15	500	200	10	20
16	500	200	10	20
17	500	100	10	10
18	500	200	10	10
19	1000	300	10	10
20	1000	400	10	10
21	300	500	20	20
22	500	600	20	20
23	1000	400	20	30
24	1000	400	20	30

2.5.4 ITC-07 (Track 3) Benchmark Dataset

The CB-CTT variant dataset (21 instances) for International Timetabling Competition (ITC-07) can be downloaded from <http://www.cs.qub.ac.uk/itc2007/index.htm>. In the last 10 years, seven different approaches were proposed for this dataset. A network flow methodology (GRASP + SA) (Kampke et al., 2019) is superior than ATS (Lü and Hao, 2010). IP relaxation proposed by Bagger et al. (2019) outperformed the other six methodologies by improving the lower bounds for three of the problem instances. The features of the dataset are given in Table 2.7.

Table 2.7: The Features of the ITC-07 (Track 3) Benchmark Dataset

Instances	Rooms	Courses	Curriculum	Constraints
01	6	30	14	53
02	16	82	70	513
03	16	72	68	382
04	18	79	57	396
05	9	54	139	771
06	18	108	70	632
07	20	131	77	667
08	18	86	61	478
09	18	76	75	405
10	18	115	67	694
11	5	30	13	94
12	11	88	150	1368
13	19	82	66	468
14	17	85	60	486
15	16	72	68	382
16	20	108	71	518
17	17	99	70	548
18	9	47	52	594
19	16	74	66	475
20	19	121	78	691
21	18	94	78	463

2.5.5 Hard Benchmark Dataset

The 60 instances (20 smalls, 20 mediums and 20 larges) proposed by Lewis (2007) can be downloaded from the Centre for Emergent Computing website with the URL <http://www.rhydlewis.eu/hardTT/>. The current state-of-the-art method for this dataset is ILS proposed by Song et al. (2018) and TSSP-ILS proposed by Goh et al. (2020). They managed to find feasible solutions for 58 and 57 instances respectively. Table 2.8 shows the features of the dataset.

Table 2.8: The Features of Hard Benchmark Dataset

Instances	Students	Events	Rooms	Features
01	1000	1000	28	20
02	1000	1000	25	20
03	900	1000	25	20
04	800	1050	25	20
05	1000	1075	25	20
06	1000	1075	25	20
07	1100	1050	25	20
08	1000	1025	25	20
09	800	1050	25	20
10	1000	1075	25	20
11	1000	1075	25	20
12	1000	1000	26	25
13	1000	1000	25	25
14	1000	1000	25	25
15	1000	1000	25	25
16	1000	1000	25	10
17	1200	1000	25	10
18	1000	1000	25	10
19	1000	1000	25	10
20	1000	1000	25	10
21	400	400	10	10
22	400	390	10	10
23	400	390	10	10
24	400	410	10	9
25	450	410	10	9
26	450	410	11	10
27	450	410	11	10
28	400	400	10	10
29	400	400	10	10
30	500	400	10	8
31	800	400	10	8
32	800	400	10	8
33	800	400	10	8
34	1000	400	10	8
35	500	425	10	8
36	1000	400	10	8
37	800	400	10	8
38	1000	400	10	8
39	1000	410	10	8
40	1000	410	10	8
41	200	200	5	5
42	400	210	6	5
43	400	200	6	5
44	500	200	5	8

Table 2.8 continued

45	500	200	5	8
46	1000	200	5	3
47	800	200	5	3
48	1000	225	5	10
49	900	225	5	10
50	1000	220	5	10
51	1000	200	5	4
52	1000	225	5	10
53	1000	225	5	10
54	1000	225	5	3
55	900	200	5	3
56	900	200	5	3
57	900	200	5	3
58	1000	225	5	3
59	1000	225	5	3
60	1000	225	5	3

2.5.6 ITC-2019 Benchmark Dataset

The International Timetabling Competition (ITC-2019) is the latest timetabling competition. Its benchmark dataset (30 instances) can be downloaded from <https://www.itc2019.org/>. Student sectioning is considered in these problem instances. Table 2.9 shows the features of the dataset.

Table 2.9: The Features of the ITC-2019 Benchmark Dataset

Instances	Size (MB)	Courses	Classes	Rooms	Students
01	14.55	340	1239(543 fixed)	80	1641
02	5.82	272	1852(332 fixed)	44	2116
03	3.88	353	983(79 fixed)	62	3018
04	12.60	1206	2641(530 fixed)	214	0
05	2.94	544	882(63 fixed)	90	3666
06	1.41	228	575(128 fixed)	35	1543
07	1.48	226	561(191 fixed)	44	865
08	15.92	1089	2526(1132 fixed)	70	2938
09	4.69	687	1001(318 fixed)	75	27018
10	1.94	36	711(74 fixed)	15	0
11	11.18	406	1144(97 fixed)	84	2254
12	10.74	234	460(2 fixed)	39	1988

Table 2.9 continued

13	3.03	313	487(3 fixed)	73	0
14	1.39	186	516(63 fixed)	35	1469
15	7.70	116	650(32 fixed)	29	395
16	6.71	881	1515(159 fixed)	83	3443
17	3.53	404	782(41 fixed)	67	2293
18	2.82	212	1061(115 fixed)	84	13497
19	32.11	2839	8813(2809 fixed)	768	38437
20	2.05	91	417(14 fixed)	28	821
21	42.84	1363	5081(341 fixed)	327	6925
22	4.16	357	1083(97 fixed)	63	2921
23	12.31	1290	2782(838 fixed)	208	0
24	3.10	328	502(8 fixed)	73	0
25	2.49	540	951(186 fixed)	93	5051
26	1.36	188	535(60 fixed)	36	1685
27	11.52	515	1623(443 fixed)	33	1152
28	27.25	1635	3717(312 fixed)	86	5651
29	10.83	1154	2798(449 fixed)	224	35213
30	1.74	44	676(47 fixed)	18	0

2.6 Algorithm for the Approaches/methodologies in UCTTP

In this section, we provide the algorithm for approaches/methodologies in UCTTP.

2.6.1 Variable Neighbourhood Descent

Variable neighbourhood descent (VND) is regarded as one of the best improvement local searches (Hansen et al., 2018). VND requires no parameter tuning, which is needed by some metaheuristics algorithms such as simulated annealing (SA). The strategy applied in VND to escape from local optima is by exploring different NS. There are two types of VND algorithms, namely the random VND and the sequential VND. When sequential VND is used, the algorithm walks through all the neighbourhood structures (NS) in sequential order. Whereas, for random VND, the algorithm walks through all the neighbourhood structures (NS) randomly.

Figure 2.4 shows the basic algorithm for VND (adapted from Borchani et al. (2017)). Any optimisation technique's efficacy may vary depending on the nature of the problem, how it was implemented and the features of the problem instances. However, VND has a few advantages. In ensuring a better exploration of the solution space, VND makes use of different NS to escape from local optima. This helps in finding a better quality solutions. VND accelerates its search for better quality solutions by continuously searching inside several neighbourhoods. The UCTTP requirements differ across academic institutions as policies and regulations are unique to each institution. By using VND, different NS can be implemented by taking into account specific constraints of the UCTTP for each institutions. VND also can be implemented easily because this method only selects the NS and applying the NS sequentially or randomly without the need for any parameter tuning.

```

PROCEDURE VND( )
Input neighbourhood structures  $N_k, k=1,2, \dots, n-1, n$ 
 $k \leftarrow 1$ 
 $currentSolution \leftarrow initialSolution$ 
 $penalty \leftarrow \infty$ 
REPEAT
  FOR each  $e$ 
    FOR each  $timeslot$ 
      FOR each  $venue$ 
         $candidateSolution \leftarrow currentSolution$ 
        IF feasible ( $e, timeslot, venue, N_k$ )
           $candidateSolution \leftarrow \text{move} (e, timeslot, venue, N_k)$ 
          IF  $f(candidateSolution) < penalty$ 
             $penalty \leftarrow f(candidateSolution)$ 
             $bestSolution \leftarrow candidateSolution$ 
          END IF
        END IF
      END FOR
    END FOR
  END FOR
  IF  $f(bestSolution) < f(currentSolution)$ 
     $currentSolution \leftarrow bestSolution$ 
  END IF
   $penalty \leftarrow \infty$ 
END FOR
IF there is improvement obtained
   $k \leftarrow 0$ 
ELSE
   $k \leftarrow k+1$ 
END IF
UNTIL  $k=n$ 
END PROCEDURE

```

Figure 2.4: Variable Neighbourhood Descent Algorithm

2.6.2 Tabu Search

Tabu search (TS) uses a tabu list to avoid being stuck in local optima. Whenever it is trapped in local optima, the search continues with non-improving moves, but previously solutions generated will be rejected with the use of a tabu list. The number of iterations during which reversal moves are still prohibited is determined by the tabu tenure. The tabu tenure is represented by the length of the tabu list. Figure 2.5 shows the basic algorithm for TS (adapted from Nagata (2018)).

```
PROCEDURE TabuSearch ( )  
  
  currentSolution  $\leftarrow$  initialSolution  
  bestSolution  $\leftarrow$  currentSolution  
  
  DO  
    currentSolution  $\leftarrow$   $\arg \min_{x \in N(\text{currentSolution})} f(x)$ ,  $x$  is non-tabu  
    IF currentSolution < bestSolution  
      bestSolution  $\leftarrow$  currentSolution  
    END IF  
    UPDATE the current move in TABULIST  
  LOOP UNTIL stopping condition  
  
END PROCEDURE
```

Figure 2.5: Tabu Search Algorithm

2.6.3 Simulated Annealing

Simulated Annealing (SA) accepts three types of moves. It accepts moves that are better compared to current solution, moves that are equivalent to the current solution and moves that are worse compared to current solution based on certain criteria. By accepting moves that are worse compared to current solution, it helps to escape from local optima. One of the disadvantages of SA is the requirement for parameter tuning. Figure 2.6 shows the basic algorithm for VND (adapted from Goh et al. (2017)).

```

PROCEDURE SimulatedAnnealing ( )

currentSolution  $\leftarrow$  initialSolution
T  $\leftarrow$  initial temperature
L  $\leftarrow$  initial length

DO
    FOR n = 1 to L
        candidateSolution  $\in N(\textit{currentSolution})$ 
        IF  $f(\textit{candidateSolution}) \leq f(\textit{currentSolution})$ 
            currentSolution  $\leftarrow$  candidateSolution
        ELSE
            IF  $\exp(f(\textit{currentSolution}) - f(\textit{candidateSolution}) / t) > \textit{random}(0,1)$ 
                currentSolution  $\leftarrow$  candidateSolution
            END IF
        END IF
    END FOR
    UPDATE L
    UPDATE T
LOOP UNTIL stopping condition

END PROCEDURE

```

Figure 2.6: Simulated Annealing Algorithm

2.6.4 Genetic Algorithm

Genetic algorithm (GA) tackles optimisation problems using the concept of biological evolution. In each iteration, the algorithm selects solutions from the population using selective pressure to bias it towards choosing the best members as parents and these parent solutions are used to generate children solutions for the next generation (Harada and Alba, 2020). This process is iterated until an optimal solution is generated or the time allowed has expired. In each iteration, the algorithm conforms to selection rules, crossover rules and mutation rules. Selection rules manage the selection of parents from the current population. Crossover rules are ways that parents are combined in generating children solutions for the next generation. Mutation is the way solutions are randomly changed to motivate diversity in the population. Figure 2.7 shows the basic algorithm for GA (adapted from (Harada & Alba, 2020)).

```

PROCEDURE GeneticAlgorithm ( )
GENERATE a group of initialSolutions (population)
DO
    SELECTION
    Crossover
    MUTATION
    REPLACEMENT
LOOP UNTIL stopping condition
END PROCEDURE

```

Figure 2.2: Genetic Algorithm

2.7 Limitations of the Approaches/methodologies in UCTTP

Each approach has its unique advantages and limitations. Operational research (OR) based techniques such as constraint logic programming and graph colouring are effective in generating feasible solutions but are lacking in producing good quality solutions compared to other approaches (Burke & Petrovic, 2002). Moreover, some of the early heuristics are not efficient in solving large problems (Burke et al., 2007).

Single solution-based meta-heuristics such as SA, are effective in achieving high-quality solutions. However, researchers need to consider parameter tuning when choosing meta-heuristic approaches. Researchers are working on designing an optimization algorithm that is not only effective but requires less manual parameter setting (Goh et al., 2020; Goh et al., 2019; Goh et al., 2017).

Population-based meta-heuristics such as GA, PSO and ACO are superior compared to others in terms of solution space exploration (Teoh et al., 2015). However, one of the

drawbacks of these approaches is the computational time required in finding good quality solutions.

Due to the limitation of meta-heuristic approaches which require intensive parameter tuning, hyper-heuristic approaches were introduced (Ross & Marin-Blazquez, 2005). Hyper-heuristics are general, simple and fast algorithms applicable to a variety of problem domain and can even adapt to different instances of a given benchmark dataset. Hyper-heuristics are heuristics to choose heuristics (algorithms), working on a search space of heuristics (algorithms) instead of a search space of solutions (Burke et al., 2013). The challenges in hyper-heuristic approaches are balancing information exchange and maintaining a problem domain barrier between the low-level heuristics and the high-level search methodology (Drake et al. , 2020).

Addressing multiple objectives is often challenging when tackling an optimization problem. This is because when the number of objectives increases, the proportion of non-dominated solutions in a randomly chosen set of objective vectors becomes exponentially large (Deb & Jain, 2014). Multi-criteria/ objective approaches gather much interest from researchers in finding the optimal Pareto front, a set of optimal compromise solutions. However, a large computational effort is often required in finding the Pareto front, even more so when it is desirable to have the solutions evenly spread along the Pareto fronts (Lindahl et al., 2018).

It is believed that population-based approaches and local search algorithms are suited in solution space exploration and exploitation respectively. Therefore, attempts have been made to achieve the synergy of both capabilities required in addressing optimization

problems. However, hybrid methods are more complex to implement and require greater computational costs (Ilyas & Iqbal, 2015).

2.8 Research Opportunities in UCTTP

Addressing the UCTTP is important for academic institutions yet challenging due to the size and the number of hard and soft constraints involved.

Heuristic approaches help to generate feasible solutions in reasonable computational time but they are lacking compared to meta-heuristic approaches in terms of optimization (Burke & Petrovic, 2002). Pillay and Özcan (2019) noted that constructive heuristics are important in addressing combinatorial optimization problems as they are usually used to create initial solutions which would then be improved by other approaches.

Local search (Single solution-based meta-heuristics) is promising as it is easy to implement and capable of addressing large-sized problems in reasonable computational times (Burke & Bykov, 2017). It will be interesting to test relatively new meta-heuristics such as grey wolf optimizer (GWO) (Mirjalili et al., 2014) and elitist self-adaptive step-size search (ESASS) (Azad & Hasançebi, 2014) in addressing the UCTTP. The GWO algorithm was inspired by the leadership hierarchy and hunting mechanism of grey wolves in nature. Meanwhile, ESASS was first utilized in steel frame structural design optimization. Both of these algorithms provided very competitive results in their respective problem domains.

The hybridization of approaches appears to be the best methodology to adopt. It has shown good quality results in previous research (Ilyas & Iqbal, 2015). These earlier findings have been validated by more recent work (Akkan & Gülcü, 2018). Hybrid methodologies are suitable for exploiting the strength of individual approaches.

As UCTTP is unique across institutions due to policies and regulatory requirements, it is difficult if not impossible, to compare solution approaches objectively (Gülcü & Akkan, 2020). This has led to the introduction of international timetabling competitions. The winners are selected based on the quality of the solution (evaluated by a cost function) (Ahmed et al., 2015). A solution that violates any hard constraints is considered worthless (Lewis, 2008). Recently, the benchmark dataset for the International Timetabling Competition 2019 (ITC-19) has been made available to the public. It is interesting to justify the performance of methodologies using this dataset.

Benchmark and real-world UCTTP vary in terms of the number and type of hard and soft constraints. Benchmark UCTTP are usually oversimplified and meant for objective comparison of methodologies. Meanwhile, real-world UCTTP focus on the practicality of solution at academic institutions. Even real-world UCTTP vary between them in terms of requirements due to different policies, education systems and cultures. Therefore, a general solution (particularly the underlying mathematical model) that fits all does not exist. Oude Vrielink et al. (2019) conducted a systematic review on practices in timetabling in higher education. The aim is to identify the gap (similarities and differences) between theory in timetabling problems and the practicality in real-world environments of higher education institutions. McCollum (2006) highlighted the importance of generating robust and flexible techniques that can cope with complexities that arise in real-world implementations. It is imperative to design an optimization algorithm that is not only effective but simple to use and adaptable to a range of real-world UCTTP. This will allow researchers to adapt/adopt the implementation of state-of-the-art methods on real-world UCTTP at academic institutions.

In real-world UCTTP, student sectioning is a method to improve room utilization and timetable feasibility, especially for large-sized problems. Schindl (2019) investigated student sectioning intending to divide students into optimal sections. The optimal number of sections for each course depends on various factors such as room capacity, institution budget and pedagogical constraints. Currently, the author managed to achieve optimal sectioning with equal section size. Optimal sectioning with non-equal section size is an open research issue.

2.9 Research Gap Analysis

As UCTTP is unique across institutions due to policies and regulatory requirements, it is difficult if not impossible, to find a general algorithm that can solve all the UCTTP faced by each academic institution. Benchmark UCTTP are usually oversimplified and meant for objective comparison of methodologies. Meanwhile, real-world UCTTP focus on the practicality of solution at academic institutions. All academic institutions vary between them in terms of constraints that need to be satisfied. So, this study focuses on applied research instead of fundamental research.

From the scientific literature, researchers are more interested in the operational than the strategic perspective of the timetabling problem. The operational level refers to the allocation process of lectures to rooms and time slots. Meanwhile, the strategic level refers to management decisions such as room number and capacity. Lindahl et al. (2018) conducted research from the aspect of strategic planning of academic institutions. The authors focused on two strategic components, namely the number of rooms required and the available time slots. This study aims to reduce the number of timeslots required to generate a feasible solution.

2.10 Conclusion

The university course timetabling problem is an active and important research area based on the sizeable amount of papers found in the scientific literature. The introduction of international timetabling competitions continues to motivate research in this area. This chapter highlights the approaches in addressing the university course timetabling problem (benchmark and real-world) proposed in the last 10 years. The approaches are classified according to category. In addition, they are sorted chronologically according to publication year thus giving an overview of the current trend in this domain. Features of benchmark datasets are detailed in tabular form. The origin, links, state of the art methodologies for each dataset are presented. In addition, this chapter provides limitations of each category of methodologies. The research opportunities in relation to the university course timetabling problem are also discussed.

CHAPTER 3

CASE STUDY

3.1 Introduction

This chapter highlights the description of the UCTTP at the FCSIT. The formal representation of the UCTTP using mathematical modelling is also presented.

Two real semester datasets were gathered from the faculty's administrator/ timetable planner. Table 3.1 shows the data instances used as testbeds. In this study, all the individual courses are referred to as events. The instance of semester 2 is smaller than that of semester 1 in terms of events, students and course enrolment because third-year students undergo industrial training.

Table 3.1: FCSIT Data Instances for Academic Years 2019/2020

Instance	Events	Rooms	Students	Timeslot requirement	Event enrolment
Semester 1 2019/2020	100	19 (fixed) 4 (shared)	1454	31	5168
Semester 2 2019/2020	77	18 (fixed) 2 (shared)	1040	31	3394

3.2 Problem Description

UNIMAS, a public university in Malaysia, was established on 24 December 1992. It has 10 faculties offering more than 90 programs. The timetabling problem in this study is based on the real-world scenario at the Faculty of Computer Science and Information Technology (FCSIT), Universiti Malaysia Sarawak (UNIMAS).

The academic year consists of two semesters and the timetables for these two semesters are scheduled independently. Meanwhile, each faculty's administrator/timetable planner at UNIMAS constructs a course timetable manually based on the curriculum (as information on course pre-registration is unavailable). The planners started utilising commercial timetabling software in 2014. The timetables are developed based on the list of courses specified in the course plan. Therefore, a timetable cannot cater to "repeaters" whose course list registrations have deviated from the original course plan.

3.2.1 Curriculum-based Course Timetabling

The five undergraduate programs offered by FCSIT include Network Computing (NC), Information Systems (IS), Multimedia Computing (MC), Computational Science (CS) and Software Engineering (SE). All five undergraduate programs are four-year programs. Students spend seven semesters on campus and one semester in industrial training. At the time of writing, the FCSIT is practising curriculum-based course timetabling, which means the timetable is generated based on the course plan for each program.

Each program has its course plan, which includes the list of courses for which students must register each semester to graduate. The courses offered by the FCSIT can be divided into various categories, namely lectures, lectures with tutorials and lectures with lab sessions. These courses range from two to four credit hours. A credit hour indicates the number of lecture hours per week for a course. It is recommended to split long lecture hours (four-credit-hour courses) into two days, for example, two hours on Monday and another two hours on Wednesday, which can be represented as "2+2". Periodically, the program structure is reviewed to improve the program quality. This affects the course plan, which in turn has an impact on the timetable generated.

Even though there is a course plan that students can use as a guideline, some students might deviate from the course plan if they have failed certain courses. Students cannot register according to the course plan if they failed a course in a previous semester for which they first need to register if it is a pre-requisite for the current semester courses. Furthermore, some students might be unable to register according to the course plan if their failed courses clash with the current courses. This group of students, which is classified as “repeaters”, will have a different course registration list compared to those who managed to pass all the courses on which they had previously registered. The current timetable used by the FCSIT cannot cater to all the “repeater” students because each “repeater” is a unique case. For example, one student may need to repeat three courses from a previous semester, whereas another may only need to repeat one. In the worst case, a student may need to repeat courses from two or three previous semesters. As a consequence, the “repeaters” may need to extend the duration of their study and be unable to graduate on time. Other consequences are that the financial cost and duration of the study are increased. On the other hand, the current timetable has also become highly rigid because of the “repeaters”. For example, the timetable for the entire cohort of an intake may have to be adjusted because of one “repeater”.

3.2.2 Timeslots

Figure 3.1(a) shows the ideal timeslots used in this study. A total of 31 timeslots are allocated for the courses, with the latest being 5 pm on Monday, Tuesday and Thursday, and noon on Friday. Figure 3.1(b) shows the number of timeslots required by the existing UNIMAS timetabling software to obtain a feasible solution. A total of 48 timeslots are required. As shown, some of the lectures are conducted until 9 pm. This consumes extra resources such as electricity and is not preferred by either academicians or students. The grey timeslots indicate that the timeslots are blocked. No assignment of faculty courses to

these timeslots is allowed. The intricacy of the UCTTP is substantially increased by the lack of available timeslots because the timetable is tight (since not many vacant places are available and it is difficult to satisfy the conflicting requirement). Table 3.2 shows the usage of blocked timeslots set by UNIMAS.

Day\Time	8-9	9-10	10-11	11-12	12-13	13-14	14-15	15-16	16-17
Mon	1	2	3	4	5	6	7	8	9
Tue	10	11	12	13	14	15	16	17	18
Wed									
Thurs	19	20	21	22	23	24	25	26	27
Fri	28	29	30	31					

(c) Ideal Timeslots Used in This Study

Day\Time	8-9	9-10	10-11	11-12	12-13	13-14	14-15	15-16	16-17	17-18	18-19	19-20	20-21
Mon	1	2	3	4	5	6	7	8	9	10	11	12	13
Tue	14	15	16	17	18	19	20	21	22	23	24	25	26
Wed													
Thurs	27	28	29	30	31	32	33	34	35	36	37	38	39
Fri	40	41	42	43	44	45	46	47	48				

(b) The Number of Timeslots Required by the Existing Timetabling Software

Figure 3.1: Number of Timeslots

Table 3.2: Usage of Blocked Timeslots

Blocked Timeslots	Reason
Wednesday	Allocated for elective courses
Friday afternoon	Allocated for generic courses

3.2.3 Venues

In terms of venue, the FCSIT conducts lectures at either its venue (available all the time) or a shared venue (only available at certain times). Sharing venues for lectures is a

common feature of most academic institutions, especially when venues can accommodate a large number of students. It is hard to split a course with many students when a limited number of venues (with a high seating capacity) can accommodate all the students. The shared venues assigned to each faculty are different each semester and depend on student enrolment. This will cause a lack of venues with a high seating capacity. With each semester's increasing student enrolment, this situation will worsen. Table 3.3 shows the capacities of the shared and fixed venues.

Table 3.3: Teaching Venues and their Capacity

Feature	Usage	Venue	Quantity	Capacity
Shared	Limited	DK	Vary from semester to semester	500
		BS	Vary from semester to semester	150
Fixed (Faculty)	All the time	TMM	1	120
		MM2	1	100
		ARTLNT	1	80
		ISLAB	1	80
		MM1	1	80
		TL1	1	80
		TL2	1	80
		CSLAB	1	60
		NETLAB1	1	60
		NETLAB2	1	40
		TR	8	40

3.2.4 Pre-registration Data Practice

At UNIMAS, there is no student pre-registration. Student course registration only takes place after the timetable has been generated. Therefore, the timetable generated cannot cater to all the students individually.

Student enrolment data is predicted by the program coordinators by gut feeling. There are uncertainties when preparing the student-course registration data. Eventually, venue utilisation is low, especially since a large-capacity venue may be allocated but a small number of students register. Other than that, “critical courses” involving “graduating students” or final-year students cannot be identified, which causes drastic changes to the planned timetable.

3.2.5 Size of Problem Instance

The complexity of a timetabling problem will increase further when the intake size at the FCSIT increases. This is because when the number of students increases, a feasible timetable will be needed to satisfy more constraints. For example, the number of students for semester 1 2019/2020 was 1,454, with a total of 5,168 enrolments, whereas for semester 2 2019/2020 the number was only 1,040, with a total of 3,394 enrolments including repeaters.

The number of repeaters can significantly increase the complexity of the timetabling problem at FCSIT. Repeaters have limited choices when registering for courses because they need to retake certain courses which different from the planned course structure for their cohort. The timetable generated must take into consideration their unique requirements to avoid clashes with their cohort schedule. Repeaters also need to avoid clashes between repeaters themselves. Flexibility in allocating their courses is limited due to the need to

satisfy constraints such as curriculum guidelines, lecturer availability and venue availability. When considering generating the timetable involving the repeaters, iterations and modifications are required. The iterative nature of the problem makes the calculation more difficult and necessitates the use of effective algorithms or heuristics to create workable and effective solutions. Changes made to accommodate repeaters also influence the timetables of other students.

3.3 Modelling and Formulation

As mentioned earlier, the university course timetabling problem (UCTTP) involves allocating a set of courses to limited resources - namely lecturers, venues and timeslots - by fulfilling certain constraints. In this study, the aim was to find a feasible timetable with the minimum number of soft constraint violations.

3.3.1 Entities

The UCTTP at the FCSIT involves the basic entities stated below:

(i) Days and timeslots

There are five working days per week and each day is divided into several timeslots. The timeslots are equal in size, at one hour long. In this case study, only four days were taken into consideration since Wednesday is used for elective course allocation.

(ii) Curriculum, courses and lectures

The curriculum consists of several courses, each of which consists of several lectures, depending on the credit hours for that course. The different curricula have different common groups of students. All the course lectures must be assigned to unique timeslots to avoid conflict. Only the timeslots available for allocation can be assigned to lectures, and the

lectures should be split into a minimum number of working days to improve the quality of the timetable. Curriculum information can be obtained from the Undergraduate Guidebook, which is published for each annual intake.

(iii) Lecturers

Each course is assigned to specific lecturers, so different lectures taught by the same lecturer cannot be assigned to the same timeslot. In this study, each course has one lecturer.

(iv) Venues

Each venue has a certain seating capacity, which determines how many students can be accommodated in the room. The facilities in the rooms determine the room's suitability for a particular course.

3.3.2 List of Hard Constraints and Soft Constraints

The list of hard constraints is as stated below:

HC_1 : A lecturer conducts at most one course per timeslot.

HC_2 : A course can be assigned to a venue and a timeslot.

HC_3 : A classroom's capacity must be sufficient to accommodate the number of students enrolled in the course.

HC_4 : All the courses must be scheduled.

HC_5 : Blocked timeslots are prohibited for course allocation.

HC_6 : A student can only attend one course per timeslot.

The list of soft constraints is as stated below:

SC_1 : Courses are split over a minimum number of working days.

SC_2 : Students should have no more than five continuous lecture hours.

SC_3 : Students should not have isolated courses with one lecture hour.

3.3.3 Notations

Table 3.4 shows the set notations and Table 3.5 shows the parameter notations of the mathematical model used as a formal representation of the UCTTP at the FCSIT.

Table 3.4: Set Notations of the Mathematical Model

Symbol	Description
$D = \{1, 2, \dots, 4\}$	Set of days
$T = \{1, 2, \dots, 31\}$	Set of timeslots per week
$R = \{1, 2, 3, \dots, r_{ R }\}$	Set of rooms. Each room has a room capacity specified by P_r . $ R $ = number of available rooms
$V_{r,t} = \{v_{r,t} : r \in R \text{ and } t \in T\}$	Set of availability for room r in timeslot t . $V_{r,t} = \begin{cases} 1 & \text{if room } r \text{ is available in timeslot } t \\ 0 & \text{otherwise} \end{cases}$ for each $r \in R$ and $t \in T$
$C = \{c_1, c_2, \dots, c_{ C }\}$	Set of courses
$S = \{s_1, s_2, \dots, s_{ S }\}$	Set of students
$LE = \{le_1, le_2, \dots, le_{ LE }\}$	Set of lecturers
AC_{le}	Set of courses taught by lecturer le
$A_{s,c} = \{a_{s,c} : s \in S \text{ and } c \in C\}$	Set of course c attended by student s $A_{s,c} = \begin{cases} 1 & \text{if student } s \text{ attends course } c \\ 0 & \text{otherwise} \end{cases}$ for each $c \in C$ and $s \in S$

Table 3.4 continued

$E_{c,r} = \{e_{c,r} : c \in C \text{ and } r \in R\}$	Set of course c that can be allocated to room r determined by the size of course c $E_{c,r} = \begin{cases} 1 & \text{if size of course } c \leq P_r \\ 0 & \text{otherwise} \end{cases}$ for each $c \in C$ and $r \in R$
--	---

Table 3.5: Parameter Notations of the Mathematical Model

Symbol	Description
p^{DAYS}	penalty for a course which is not split over minimum working days
p^{CONT}	penalty for students with more than five lecture hours consecutively
p^{ISO}	penalty for students with an isolated course having one lecture hour on a day

D denotes the set of days for the timetable. Each day is divided into timeslots, and the set of timeslots is denoted by T . In each timeslot, a different set of rooms R is available because some rooms are shared between the faculties. The set of courses is denoted by C . Each course c has l_c lectures that must be scheduled. These lectures should be split over a minimum number of working days specified by m_c . For example, courses with four hours of lectures must be split over a minimum of two working days. p^{DAYS} , p^{CONT} and p^{ISO} refer to the penalties for a course that is not split over a minimum number of working days, for students with more than five consecutive lecture hours and for students with an isolated course with one lecture hour on one day, respectively.

i. Decision Variables

The timetable is represented by the binary decision variable $x_{c,t,r}$, which takes the value 1 if a lecture in course c is scheduled at timeslot t in room r , provided that $V_{t,r} = 1$. The decision variables used in the mathematical modelling are defined as follows.

$$x_{c,t,r} = \begin{cases} 1 & \text{if course } c \text{ is scheduled at timeslot } t \text{ in room } r; \\ & \text{provided } V_{t,r} = 1 \text{ for each } t \in T, c \in C \text{ and } r \in R \\ 0 & \text{otherwise} \end{cases} \quad \text{Equation 3.1}$$

$$z_{c,d} = \begin{cases} 1 & \text{if } \sum_{t=(d-1) \times 9 + 1}^{d \times 9, \max=31} \sum_{r \in R} x_{c,t,r} \geq 1 \text{ for each } c \in C \text{ and } d \in D \\ 0 & \text{otherwise} \end{cases} \quad \text{Equation 3.2}$$

If $z_{c,d} = 1$, it means that course c is scheduled on day d .

$$y_{s,t} = \begin{cases} 1 & \text{if } x_{c,t,r} \cdot A_{s,c} = 1 \text{ for each } r \in R \text{ and } t \in T \\ 0 & \text{otherwise} \end{cases} \quad \text{Equation 3.3}$$

If $y_{s,t} = 1$, it means that student s attends course c , which is scheduled at timeslot t in room r .

ii. Soft Constraints

The formal representations of the soft constraints are as follows:

SC_I : Courses are split over a minimum number of working days (there is a penalty if a course is not split over a minimum number of working days).

$$\sum_{c \in C} EV_c \quad \text{Equation 3.4}$$

where:

$$EV_c = \begin{cases} 1 & \text{if } \sum_{d \in D} z_{c,d} < m_c \text{ for each } c \in C \\ 0 & \text{otherwise} \end{cases} \quad \text{Equation 3.5}$$

The value for EV_c for each course is 1 if the total number of days scheduled for course c is less than m_c , the minimum split of working days for course c . Otherwise, the value for EV_c will be 0, which means the number of days scheduled for course c is equal to or more than m_c , the minimum split of working days for course c .

SC_2 : Students should have no more than five continuous lecture hours (there is a penalty if students have more than five consecutive lecture hours).

$$\sum_{s \in S} \sum_{d=1}^3 C_{s,d} \quad \text{Equation 3.6}$$

where:

$$C_{s,d} = \sum_{t=(d-1) \times 9 + 1}^{d \times 9 - 5} y_{s,t} \cdot y_{s,(t+1)} \cdot y_{s,(t+2)} \cdot y_{s,(t+3)} \cdot y_{s,(t+4)} \cdot y_{s,(t+5)} \quad \text{Equation 3.7}$$

for each $s \in S$ and $d \in D$

$C_{s,d}$ is used to store the information on a student, s , who has more than five consecutive lecture hours. For example, if on day d , student s has six consecutive lecture hours, then the value for $C_{s,d}$ is 1. However, if on day d , student s has seven lecture consecutive hours, then the value for $C_{s,d}$ is 2. The number of days is three because Wednesday is used for elective course allocation and Friday only has four lecture hours.

SC_3 : Students should not have isolated courses with one lecture hour (there is a penalty if students have an isolated course with one lecture hour on one day).

$$\sum_{s \in S} \sum_{d=1}^4 I_{s,d} \quad \text{Equation 3.8}$$

where:

$$I_{s,d} = \begin{cases} 1 & \text{if } \sum_{t=(d-1) \times 9 + 1}^{d \times 9, \max=31} y_{s,t} = 1 \\ 0 & \text{otherwise} \end{cases} \quad \text{for each } s \in S \text{ and } d \in D \quad \text{Equation 3.9}$$

The value for $I_{s,d}$ is 1 if a student s has an isolated single lecture hour in day d . If the value for $I_{s,d}$ is 0, it means that a student s has either no lecture on that day or more than one lecture hour on day d .

The sum of the soft constraint penalties of the model is minimised by the objective function.

$$\begin{aligned} \text{MIN } \sum_{c \in C} EV_c \cdot p_{DAYS} + \sum_{s \in S} \sum_{d=1}^3 C_{s,d} \cdot p_{CONT} \\ + \sum_{s \in S} \sum_{d=1}^4 I_{s,d} \cdot p_{ISO} \end{aligned} \quad \text{Equation 3.10}$$

iii. Hard Constraints

The formal representations of the hard constraints are as follows:

HC_1 : A lecturer conducts at most one course per timeslot.

$$\sum_{c \in AC_{le}, r \in R} x_{c,t,r} \leq 1 \quad \text{for each } t \in T \text{ and } le \in LE \quad \text{Equation 3.11}$$

HC_2 : Only one course can be assigned to a venue for each timeslot.

$$\sum_{c \in C} x_{c,t,r} \leq 1 \quad \text{for each } t \in T \text{ and } r \in R \quad \text{Equation 3.12}$$

HC_3 : A classroom's capacity must be sufficient to accommodate the number of students enrolled on the course.

$$E_{c,r} \cdot x_{c,t,r} = x_{c,t,r} \quad \text{for each } t \in T, c \in C \text{ and } r \in R \quad \text{Equation 3.13}$$

HC_4 : All the courses must be scheduled.

$$\sum_{t \in T, r \in R} x_{c,t,r} = l_c \quad \text{for each } c \in C \quad \text{Equation 3.14}$$

HC_5 : Blocked timeslots are prohibited for course allocation.

The blocked timeslots that are prohibited for course allocation are not included in the set of timeslots, T . The decision variable, $x_{c,t,r}$, which takes the value 1 if a lecture in course c is scheduled at timeslot t in room r , is defined only for timeslots that are not blocked. Thus, HC_5 has been implicitly taken into account.

HC_6 : A student can only attend one course per timeslot.

$$\sum_{r \in R} x_{c,t,r} \cdot A_{s,c} \leq 1 \quad t \in T, c \in C, s \in S \quad \text{Equation 3.15}$$

3.4 Conclusion

The case study of the FCSIT timetabling problem was explored in this chapter. The problems identified with the current FCSIT timetabling method include curriculum-based course timetabling, pre-registration data practices, the size of the problem instance, the timeslots and the venues. Based on the specified hard and soft constraints, the mathematical modelling for this study is depicted.

CHAPTER 4

PROPOSED ALGORITHM: HEURISTIC ORDERINGS AND VARIABLE NEIGHBOURHOOD DESCENT

4.1 Introduction

In this chapter, the algorithms used in this study to solve the UCTTP at the FCSIT are presented. A two-stage heuristic algorithm is proposed. The algorithms for stage 1 and stage 2 will be presented in detail.

4.2 Framework for Two-stage Heuristic Algorithm

In this study, a two-stage heuristic algorithm is proposed. The UCTTP is known to be NP-hard (Wang et al., 2019; Song et al., 2018; Assi et al., 2018; Gunawan, Ng, & Poh, 2012); that is, the problem cannot be solved exactly in polynomial time as the growth of the problem size and its complexity are exponential (Babaei et al., 2015; Bardadym, 1996). By proposing two-stage heuristic, the complexity of the problem can be broken into smaller and more manageable sub-problem. In stage 1, heuristic ordering (HO), HO with a perturbation and tabu search (TS) are utilised to generate a feasible initial solution by ensuring all the hard constraints are satisfied. TS is utilised only if HO and HO with a perturbation are unable to generate a feasible initial solution. In stage 2, an improved variable neighbourhood descent (VND) is proposed to improve the quality of the solution by minimising the number of soft constraint violations. This approach does not require parameter tuning as required in metaheuristic approaches such as simulated annealing. Sequential VND is used, whereby the algorithm walks through all the neighbourhood structures (NS) in sequential order. This starts with the first NS and continues sequentially to the next. The novelties of the proposed improved VND are the implementation of four new neighbourhood structures to avoid being

stuck in local optima. Figure 4.1 shows the general framework for solving the UCTTP at the FCSIT, UNIMAS.

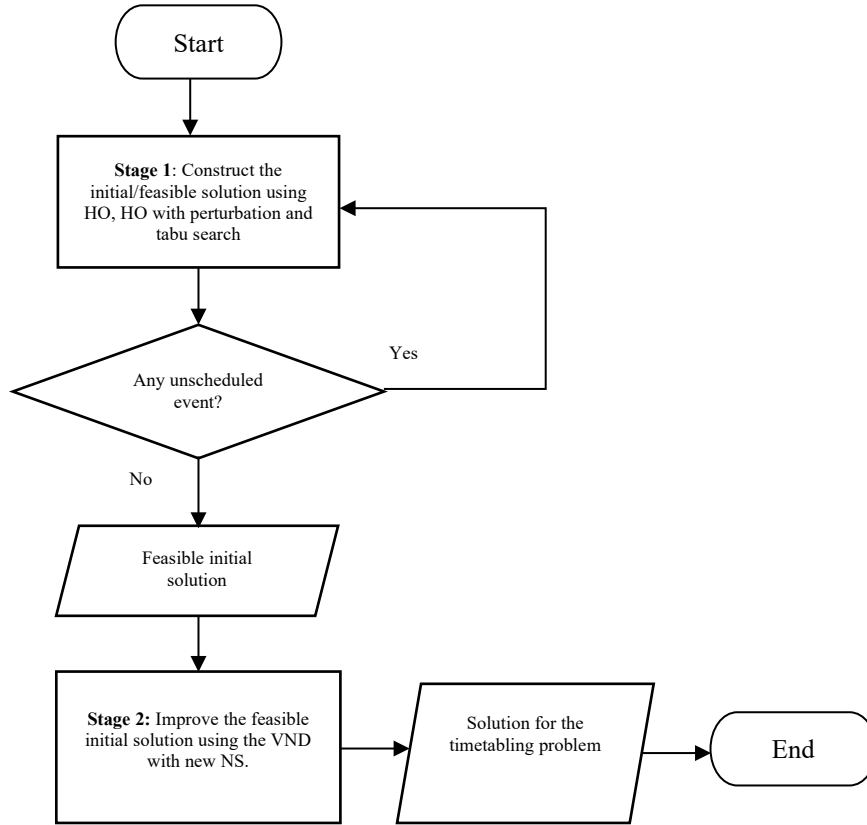


Figure 4.1: General framework for solving the UCTTP at the FCSIT, UNIMAS

4.3 Stage 1: Heuristic Ordering (HO)

The algorithms were coded using visual basic (VB.Net). Microsoft Access was used as the database management software.

4.3.1 Algorithm Description

Figure 4.2 shows the algorithm for HO used in stage 1 to generate a feasible initial solution. The types of HO used in this study are stated below:

- i. Largest Degree (LD) in descending order

In LD Descending, the event with the largest number of conflicts/clashes with other events is assigned first.

- ii. Largest Degree (LD) in ascending order

In LD Ascending, the event with the smallest number of conflicts/clashes with other events is assigned first.

- iii. Largest Enrolment (LE) in descending order

In LE Descending, the event with the largest number of enrolments is assigned first.

- iv. Largest Enrolment (LE) in ascending order

In LE Ascending, the event with the smallest number of enrolments is assigned first.

- v. (LD+LE) in descending order

In (LD+LE) Descending, the event with the largest number of conflicts/clashes with other events and the largest number of enrolments is assigned first.

- vi. (LD+LE) in ascending order

In (LD+LE) Ascending, the event with the smallest number of conflicts/clashes with other events and the smallest number of enrolments is assigned first.

- vii. Random order

In random order, no sorting is utilised. The order is based on the listing in the database.

The order of events remains constant throughout the allocation process while using any type of HO. The *orderedEvents* is a list of events ordered based on HO. If the number of conflicts/clashes with other events or the number of enrolments is the same for a group of events, then an event is selected randomly. For each event in *orderedEvents*, the timeslots and venues are searched sequentially until a feasible and available *solution* is found. Once it is found, an event *e* is assigned to the *timeslot* and *venue* identified. The next event in *orderedEvents* will then be considered. If no feasible and available *solution* can be found, the next event in *orderedEvents* will be considered.

```

PROCEDURE HO

FOR each e
  assign(e)  $\leftarrow$  0
END FOR

FOR each e in orderedEvents
  FOR each timeslot
    FOR each venue
      IF feasible_available (e, timeslot, venue)
        solution  $\leftarrow$  move (e, timeslot, venue)
        assign(e)  $\leftarrow$  1
        IF assign(e) = 1
          Break;
        END IF
      END IF
    END FOR
    IF assign(e) = 1
      Break;
    END IF
  END FOR
END FOR

END PROCEDURE

```

Figure 4.2: HO Algorithm

4.4 Stage 1: Heuristic Ordering (HO) with a Perturbation

The proposed algorithm is further improved in stage 1 in terms of finding feasibility. A perturbation is applied to HO to find a feasible initial solution. The perturbation shakes

the current solution and the allocation process will be reimplemented for the unassigned events. The current solution is shaken by rescheduling each allocated event to another available timeslot. This means all the previously assigned events will not be unassigned from the current solution. An event is only moved to another available timeslot if the move does not violate any hard constraints specified in the problem description. The objective of the perturbation is to increase the exploration of the search space.

4.4.1 Algorithm Description

Figure 4.3 shows the algorithm for HO with a perturbation used in stage 1 to generate a feasible initial solution. After an event is selected from *orderedEvents*, it is checked to see if it has been allocated in the *solution* or not. If it has, the next event in *orderedEvents* will be considered. After all the events have been processed, the procedure `STAGE1PERTURBATION(solution)` will be implemented before the next iteration. The whole process will stop after a certain number of iterations, *numberofiteration*.

```

PROCEDURE HO

FOR each  $e$ 
  assign( $e$ )  $\leftarrow$  0
END FOR

FOR  $i=0$  to numberofiteration
  FOR each  $e$  in orderedEvents
    IF assign( $e$ )= 0
      FOR each timeslot
        FOR each venue
          IF feasible_available ( $e$ , timeslot, venue)
            solution  $\leftarrow$  move ( $e$ , timeslot, venue)
            assign( $e$ )  $\leftarrow$  1
            IF assign( $e$ ) = 1
              Break;
            END IF
          END IF
        END FOR
      END FOR
      IF assign( $e$ ) = 1
        Break;
      END IF
    END FOR
  END IF
  STAGE1PERTURBATION(solution)
END FOR

END PROCEDURE

```

Figure 4.3: HO Algorithm with a Perturbation

Figure 4.4 shows the STAGE1PERTURBATION procedure implemented in stage 1 of the proposed algorithm whereas Figure 4.5 shows the procedure illustratively. The objective of perturbation is to increase the exploration of the search space. It is a process of introducing plan changes to the current solution to explore new areas of the search space which is the local neighbourhood of the current solution. The perturbation shakes the current solution and the allocation process will be reimplemented for the unassigned events. As shown in Figure 4.5, TMS4013 is unallocated. So, perturbation is implemented by reallocating TMF1814, TMI3053 and TME2104 to another available timeslots. This means all the previously assigned events e will not be unassigned from the *solution*. An event is

only moved to the available timeslot if the move does not violate any hard constraints specified in the problem statement. An event is selected from *LEorderedEvents* (in which events are sorted by largest enrolment in ascending order, so the event with the fewest students is rescheduled first). If the number of enrolments is the same for a group of events, the event is selected randomly. LE is used in the STAGE1PERTURBATION procedure to ensure the low-capacity venues are utilised in the event allocation. The timeslots and venues are searched sequentially until another feasible and available *solution* has been found. Once this occurs, event e is assigned to another available *timeslot* and *venue* that have been identified but without violating any hard constraints. Then, the next event in *LEorderedEvents* is considered. If no available *timeslot* and *venue* can be found which do not violate the hard constraints, the next event in *LEorderedEvents* will be considered. This means that for the event that cannot be reallocated to another available and feasible timeslot and venue, it will be maintained in the current timeslot and venue.

```
PROCEDURE STAGE1PERTURBATION(solution)
```

```
  FOR each e in LEorderedEvents
```

```
    moved  $\leftarrow$  false
```

```
    IF assign(e)= 1
```

```
      FOR each timeslot
```

```
        FOR each venue
```

```
          IF feasible_available (e, timeslot, venue)
```

```
            solution  $\leftarrow$  move (e, timeslot, venue)
```

```
            moved  $\leftarrow$  true
```

```
            IF moved = true
```

```
              Break;
```

```
            END IF
```

```
          END IF
```

```
        END FOR
```

```
      IF moved = true
```

```
        Break;
```

```
      END IF
```

```
    END FOR
```

```
  END IF
```

```
END FOR
```

```
END PROCEDURE
```

Figure 4.4: Perturbation Procedure for HO Algorithm

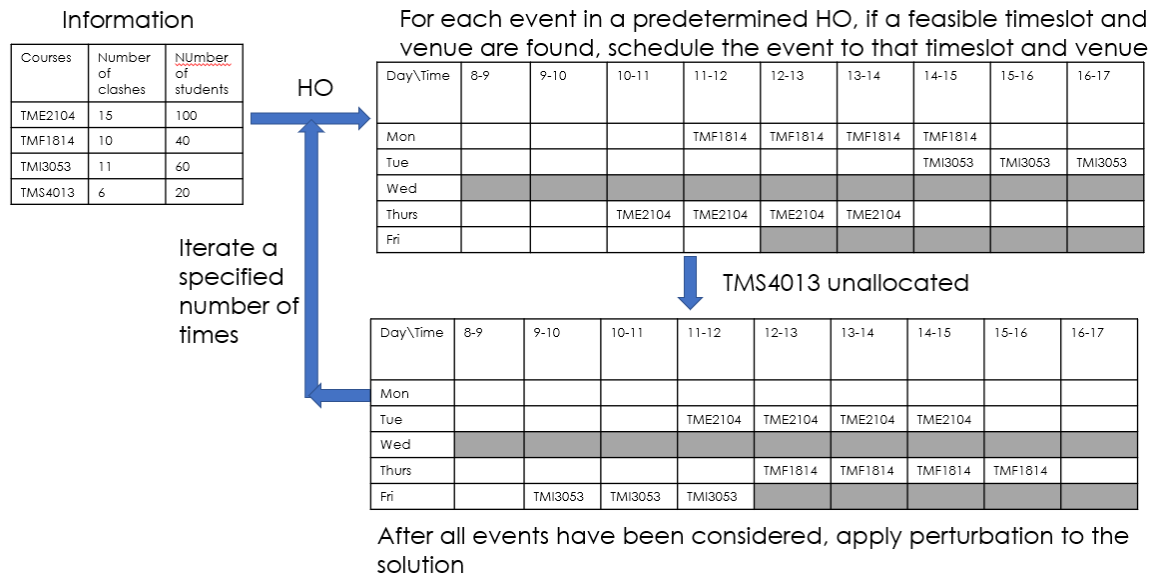


Figure 4.5: Perturbation Procedure for HO Algorithm in Diagram

4.5 Stage 1: Tabu Search (TS)

Tabu search (TS) is applied to the initial solution produced via HO with a perturbation for data instances that failed to produce a feasible initial solution after utilising HO and HO with a perturbation.

4.5.1 Algorithm Description

Tabu search (TS) uses a tabu list to avoid being stuck in local optima. Whenever it is trapped in local optima, the search continues with non-improving moves, but previously solutions generated will be rejected with the use of a tabu list. In this study, TS was utilised when HO with a perturbation did not manage to generate a feasible initial solution.

Figure 4.6 shows the TS algorithm proposed in stage 1 to generate a feasible initial solution. In this algorithm, an unassigned event, e is selected from *orderedEvents*. All neighbourhood moves are reviewed at the start of each iteration by assessing all the non-tabu acceptable timeslots and venues for the unassigned event, e .

An e is assigned to the *candidateSolution* after a feasible timeslot has been found by removing all the events clashing with e . All the possible neighbourhood moves are considered and the move with the lowest cost function is executed to form the *bestSolution*. The cost function f is used to determine the *bestSolution* from all *candidateSolution* by summing up the number of unassigned events. The *candidateSolution* with the lowest cost function is chosen as the *bestSolution*. Then, the move made to generate the *bestSolution* is placed in the tabu list before the next iteration is started. The *bestSolution* will be the *currentSolution* before the next iteration is executed. The tabu tenure length used in this study is 10 as this is proven to be the best (Goh et al., 2017). The iteration continues until all the events have been assigned to the timetable or a specified time limit has been reached, which is 180 seconds for this study.

```

PROCEDURE TabuSearch (bestSolution, bestAssign)

currentSolution  $\leftarrow$  bestSolution
currentAssign  $\leftarrow$  bestAssign
unassignedEvent  $\leftarrow$   $\infty$ 

DO
  FOR each e in orderedEvents
    IF currentAssign(e) = 0
      FOR each timeslot
        FOR each venue
          IF feasible_available (e, timeslot, venue) AND non-tabu move
            candidateSolution  $\leftarrow$  currentSolution - {events_conflicting_e}
            candidateSolution  $\leftarrow$  move (e, timeslot, venue)
            candidateAssign(e)  $\leftarrow$  1
            FOR each {events_conflicting_e}
              candidateAssign(event_conflicting_e)  $\leftarrow$  0
            END FOR
            IF f(candidateSolution) < unassignedEvent
              unassignedEvent = f(candidateSolution)
              bestSolution  $\leftarrow$  candidateSolution
              bestAssign  $\leftarrow$  candidateAssign
            END IF
          END IF
        END FOR
      END FOR
    END IF
  END FOR
  unassignedEvent  $\leftarrow$   $\infty$ 
  SET tabu move
  currentSolution  $\leftarrow$  bestSolution
  currentAssign  $\leftarrow$  bestAssign
END IF
END FOR
LOOP UNTIL (checkAssignment(currentAssign) = 1 OR timespent.Totalseconds  $\geq$  180)

END PROCEDURE

```

Figure 4.6: TS Algorithm

4.6 Stage 2: Variable Neighbourhood Descent (VND)

In stage 2, a method called variable neighbourhood descent (VND) is used to minimise the number of soft constraint violations. VND is regarded as one of the best improvement local searches (Hansen et al., 2018).

4.6.1 Algorithm Description

VND is proposed to solve the UCTTP at the FCSIT because VND requires no parameter tuning, which is needed by some metaheuristics algorithms such as simulated annealing (SA). The neighbourhood structures (NS) used in the existing scientific literature are simple moves involving one event and swaps involving only two events. In the proposed algorithm, the newly suggested NS are a simple move involving two events and a swap involving three events at the same time. A total of eight NS are used in the proposed algorithm. NS1, NS2, NS5 and NS6 are the NS used in the existing scientific literature (Aladag et al., 2009), while NS3, NS4, NS7 and NS8 are the new NS implemented in the proposed algorithm.

Sequential VND is used, whereby the algorithm walks through all the neighbourhood structures (NS) in sequential order. It starts with the first NS and continues sequentially to the next.

Figure 4.7 illustrates the neighbourhood structures (NS) adopted in the proposed algorithm:

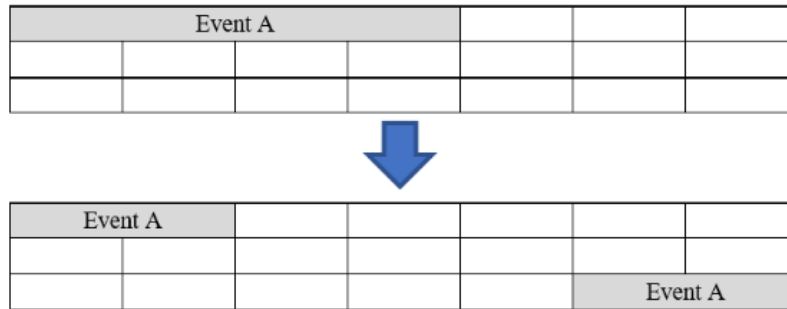
- Neighbourhood structure 1 (NS1): attempts to split a course with four continuous lecture hours by moving two of its lecture hours to other timeslots.

- Neighbourhood structure 2 (NS2): attempts to split a course with four continuous lecture hours by swapping two of its lecture hours with another course that has two lecture hours.
- Neighbourhood structure 3 (NS3): attempts to split a course with four continuous lecture hours by executing two moves (two lecture hours) involving another course.
- Neighbourhood structure 4 (NS4): attempts to split a course with four continuous lecture hours by executing two swaps (two lecture hours) involving two other courses.
- Neighbourhood structure 5 (NS5): attempts to split a course with four or three continuous lecture hours by moving one of its lecture hours to another timeslot.
- Neighbourhood structure 6 (NS6): attempts to split a course with four or three continuous lecture hours by swapping one of its lecture hours with another course that has one lecture hour.
- Neighbourhood structure 7 (NS7): attempts to split a course with four or three continuous lecture hours by executing two moves (one lecture hour) involving another course.
- Neighbourhood structure 8 (NS8): attempts to split a course with four or three continuous lecture hours by executing two swaps (one lecture hour) involving two other courses.

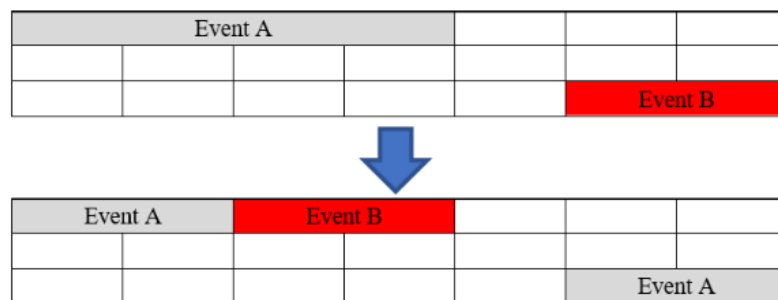
NS3, NS4, NS7 and NS8 are the newly introduced neighbourhood structures and were included in the proposed algorithm to further improve the quality of the timetable. As shown in Figure 4.7, Event A is selected from a list ordered by HO (LE) in descending order, whereas Event B and Event C are selected when the timeslots and venues are scanned sequentially. The eight different NS are used to improve the connectivity of the search space

and therefore the quality of the solution. If the resulting solution following the application of the NS is feasible (it does not breach any hard constraints), it is returned as a *candidateSolution* and evaluated for acceptance.

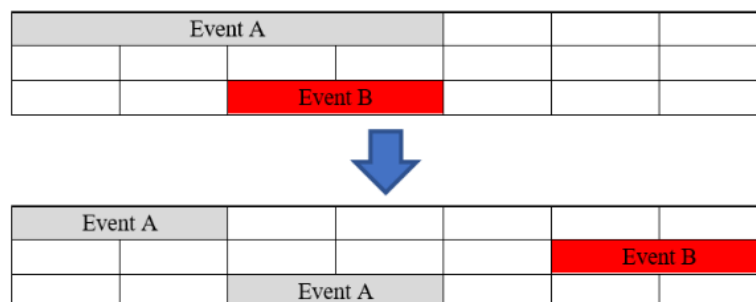
NS1: Split a course with four continuous lecture hours by moving two of its lecture hours to other timeslots.



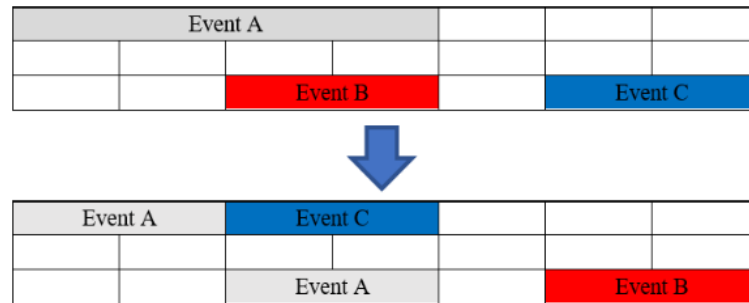
NS2: Split a course with four continuous lecture hours by swapping two of its lecture hours with another course that has two lecture hours.



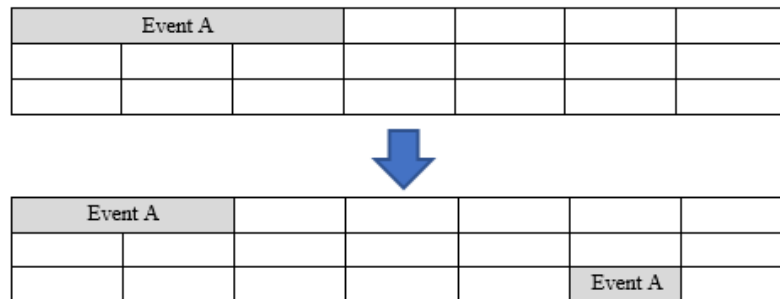
NS3: Split a course with four continuous lecture hours by executing two moves (two lecture hours) involving another course.



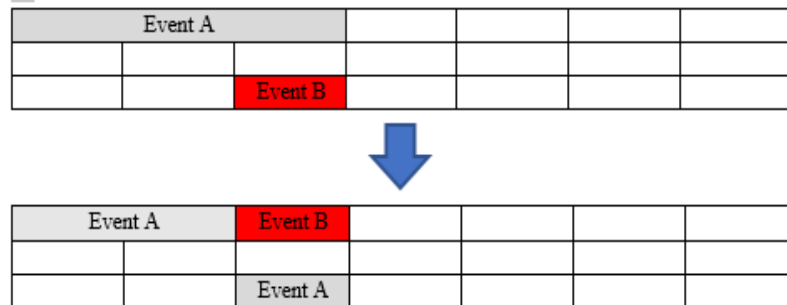
NS4: Split a course with four continuous lecture hours by executing two swaps (two lecture hours) involving two other courses.



NS5: Split a course with four or three continuous lecture hours by moving one of its lecture hours to another timeslot.



NS6: Split a course with four or three continuous lecture hours by swapping one of its lecture hours with another course that has one lecture hour.



NS7: Split a course with four or three continuous lecture hours by executing two moves (one lecture hour) involving another course.



NS8: Split a course with four or three continuous lecture hours by executing two swaps (one lecture hour) involving two other courses.



*Note: Column – timeslot, Row – venue

Figure 4.7: Neighbourhood structures: NS1 to NS8

Figure 4.8 shows the details of this algorithm. k is initialised to 1. The algorithm starts with a feasible initial solution obtained from stage 1. *orderedEvents* is a list of events ordered based on HO (LE) in descending order. For each event in *orderedEvents*, the timeslots and venues are searched sequentially until a feasible *candidateSolution* has been found. Once this occurs, the $f(\text{candidateSolution})$ and *penalty* values are compared. The cost function f is used to determine the *bestSolution* from all *candidateSolution* by summing up the number of soft constraint violations. If the value of $f(\text{candidateSolution})$ is less than the value of *penalty*, the *candidateSolution* will be set as the *bestSolution*. All the possible neighbourhood moves are considered and the one with the lowest cost function is executed to form the *bestSolution*. After the iteration, the $f(\text{bestSolution})$ and $f(\text{currentSolution})$ are compared. If the $f(\text{bestSolution}) < f(\text{currentSolution})$, the *bestSolution* is assigned as the *currentSolution*. Then, the next event in *orderedEvents* will be considered. After applying each neighbourhood structure, the improvement to the solution obtained is checked. If there is an improvement, the first neighbourhood structure is applied to the events. Otherwise, the next neighbourhood structure is applied to the events.

After satisfying the first soft constraint, the algorithm continues with the SC_2 and SC_3 . Whereas each event is iterated for the first soft constraint, for the second and third, each student is iterated to generate a better-quality solution. Figure 4.9 shows the VND algorithm applied to solve the second and third soft constraints for the real dataset as well as all the soft constraints for the ITC-2007 (Track 2) benchmark dataset. In solving the second and third soft constraints for each *studentevent*, the timeslots and venues are searched sequentially until a feasible *candidateSolution* is found. Once this has been found, the $f(candidateSolution)$ and *penalty* values are compared. If the $f(candidateSolution)$ value is less than the value of *penalty*, the *candidateSolution* will be set as the *bestSolution*. All possible neighbourhood moves are considered and the move with the lowest cost function is executed to form the *bestSolution*. After the iteration, the $f(bestSolution)$ and the $f(currentSolution)$ are compared. If the $f(bestSolution) < f(currentSolution)$, the *bestSolution* is assigned as the *currentSolution*. Then, the next student will be considered. After applying each neighbourhood structure, the improvement to the solution obtained is checked. If there is an improvement, the first neighbourhood structure is applied to the events. Otherwise, the next neighbourhood structure is applied to the events.

```

PROCEDURE VNDFIRSTSC
Input neighbourhood structures  $N_k, k=1,2,3,4,5,6,7,8$ 
 $k \leftarrow 1$ 
 $currentSolution \leftarrow initialSolution$  //initial solution is obtained from stage 1
 $orderedEvents \leftarrow$  events ordered based on HO
 $penalty \leftarrow \infty$ 
REPEAT
  FOR each  $e$  in  $orderedEvents$ 
    FOR each  $timeslot$ 
      FOR each  $venue$ 
         $candidateSolution \leftarrow currentSolution$ 
        IF feasible ( $e, timeslot, venue, N_k$ )
           $candidateSolution \leftarrow \text{move}(e, timeslot, venue, N_k)$ 
          IF  $f(candidateSolution) < penalty$ 
             $penalty \leftarrow f(candidateSolution)$ 
             $bestSolution \leftarrow candidateSolution$ 
          END IF
        END IF
      END FOR
    END FOR
  END FOR
  IF  $f(bestSolution) < f(currentSolution)$ 
     $currentSolution \leftarrow bestSolution$ 
  END IF
   $penalty \leftarrow \infty$ 
END FOR
IF there is improvement obtained
   $k \leftarrow 0$ 
ELSE
   $k \leftarrow k+1$ 
END IF
UNTIL  $k=8$ 
END PROCEDURE

```

Figure 4.8: VND Algorithm Solving $SC1$

```

PROCEDURE VNDSECONDTTHIRDSC
Input neighbourhood structures  $N_k, k=1,2,3,4,5,6,7,8$ 
 $k \leftarrow 1$ 
 $currentSolution \leftarrow initialSolution$  //initial solution is obtained from stage 1
 $penalty \leftarrow \infty$ 
REPEAT
  FOR each studentevent
    FOR each timeslot
      FOR each venue
         $candidateSolution \leftarrow currentSolution$ 
        IF feasible (studentevent, timeslot, venue,  $N_k$ )
           $candidateSolution \leftarrow move (studentevent, timeslot, venue, N_k)$ 
          IF  $f(candidateSolution) < penalty$ 
             $penalty \leftarrow f(candidateSolution)$ 
             $bestSolution \leftarrow candidateSolution$ 
          END IF
        END IF
      END FOR
    END FOR
  END FOR
  IF  $f(bestSolution) < f(currentSolution)$ 
     $currentSolution \leftarrow bestSolution$ 
  END IF
   $penalty \leftarrow \infty$ 
END FOR
IF there is improvement obtained
   $k \leftarrow 0$ 
ELSE
   $k \leftarrow k+1$ 
END IF
UNTIL  $k=8$ 
END PROCEDURE

```

Figure 4.9: VND Algorithm Solving SC2 and SC3 and ITC-2007 (Track 2) Benchmark Datasets

4.7 Stage 2: Variable Neighbourhood Descent (VND) with a Perturbation

To further improve the quality of the timetable generated, a perturbation is applied to the proposed algorithm. The perturbation shakes the current solution and the VND algorithm is implemented again to improve the quality of the solution. The iteration will stop when no improvement is obtained from the implementation of the proposed VND algorithm for all the soft constraints.

4.7.1 Algorithm Description

Figure 4.10 depicts the algorithm for VND with a perturbation. The perturbation is applied to VND until no improvement is obtained.

```
PROCEDURE STAGE2()  
   $a \leftarrow 1$   
  REPEAT  
    IF  $a$  NOT EQUAL 1  
      STAGE2PERTURBATION()  
    END IF  
     $a \leftarrow a + 1$   
    VNDFIRSTSC()  
    VNDSECONDTHIRDSC()  
  UNTIL no improvement is obtained
```

Figure 4.10: VND with a Perturbation

Figure 4.11 shows the procedure for the perturbation algorithm used in stage 2. An event is selected randomly. For each event, the timeslots and venues are searched sequentially for available timeslots and venues until a feasible solution is found. Once this has been found, the event is moved to the available timeslot and venue. Then, the next event

is considered. If no feasible solution can be found from the available timeslot and venue, the next event will be considered. This means that for the event that cannot be reallocated to another available and feasible timeslot and venue, it will be maintained in the current timeslot and venue.

```
PROCEDURE STAGE2PERTURBATION(solution)
```

```

FOR each e
  moved  $\leftarrow$  false
  FOR each timeslot
    FOR each venue
      IF feasible_available (e, timeslot, venue)
        solution  $\leftarrow$  move (e, timeslot, venue)
        moved  $\leftarrow$  true
        IF moved = true
          Break;
        END IF
      END IF
    END FOR
  IF moved = true
    Break;
  END IF
END FOR
END FOR
END PROCEDURE
```

Figure 4.11: Perturbation Procedure for the Proposed VND Algorithm

4.8 Stage 2: Tabu Search (TS) -Used as a Comparison to the Improved VND

For the second stage, the improved Variable Neighbourhood Descent (VND) is being compared with Tabu Search (TS) to evaluate their performance on the given problem. The purpose of the comparison is to provide a foundational performance benchmark for the improved VND. TS is chosen because from the literature review, a few researchers use this method in solving the UCTTP. Those who are intrigued can refer to Nagata (2018), Aladag

et al. (2009), Goh et al. (2017), Lü and Hao (2010) and Vianna et al. (2020). It has been widely used to solve benchmark datasets and real-world datasets for UCTTP.

Tabu search (TS) is categorized as single solution-based meta-heuristics. The concept of this method is to keep in memory the moves in the search space that are deemed to be “taboo” or forbidden. The forbidden moves are put in a tabu list. This tabu list helps the algorithm to avoid being stuck in local optima and explore the search space efficiently. This is the interesting aspect of TS which can be used as a comparison for the improved VND. However, TS might require some tuning to determine the tenure length for the tabu list.

The main difference between VND and TS is the strategy both algorithms have used to avoid being trapped in local optima. VND avoid being trapped in local optima by using a few NS. When it is trapped in a local optima, it will select another NS to further explore the search space. VND does not keep the information about the recent moves and the same moves might be repeated. Whereas, TS avoid being trapped in local optima by avoiding the forbidden recent moves specified in the tabu list. So, it will not be repeating the same recent moves and explore the search space in a new directions.

4.8.1 Algorithm Description

Figure 4.12 shows the TS algorithm used in stage 2 for a comparison with the improved Variable Neighbourhood Descent (VND). NS used for TS is NS1, NS2, NS5 and NS6 from the existing scientific literature (Aladag et al., 2009). The algorithm starts with a feasible initial solution obtained from stage 1. *orderedEvents* is a list of events ordered based on HO (LE) in descending order. For each event in *orderedEvents*, the timeslots and venues are searched sequentially until a feasible *candidateSolution* has been found and is not a tabu

move. If the tabu tenure length is 10, the recent 10 events that has been recently moved are forbidden to be reallocated to another timeslot and venue. Once this occurs, the $f(candidateSolution)$ and penalty values are compared. The cost function f is used to determine the $bestSolution$ from all $candidateSolution$ by summing up the number of soft constraint violations. If the value of $f(candidateSolution)$ is less than the value of penalty, the $candidateSolution$ will be set as the $bestSolution$. All the possible neighbourhood moves are considered and the one with the lowest cost function is executed to form the $bestSolution$. After the iteration, the $f(bestSolution)$ and $f(currentSolution)$ are compared. If the $f(bestSolution) < f(currentSolution)$, the $bestSolution$ is assigned as the $currentSolution$. Then, the move made is placed in the tabu list before the next iteration is started. The whole process will be iterated until no improvement is obtained. The tabu tenure length used in this study is 10. This means 10 moves which are forbidden are stored in the tabu list.

```

PROCEDURE TabuSearch ( )

currentSolution  $\leftarrow$  initialSolution
penalty  $\leftarrow$   $\infty$ 

DO
  FOR each e in orderedEvents
    FOR each timeslot
      FOR each venue
        candidateSolution  $\leftarrow$  currentSolution
        IF feasible_available (e, timeslot, venue) AND non-tabu move
          candidateSolution  $\leftarrow$  move (e, timeslot, venue)
          IF  $f(\text{candidateSolution}) < \text{penalty}$ 
            penalty  $\leftarrow$   $f(\text{candidateSolution})$ 
            bestSolution  $\leftarrow$  candidateSolution
          END IF
        END IF
      END FOR
    END FOR
  END FOR
  IF  $f(\text{bestSolution}) < f(\text{currentSolution})$ 
    currentSolution  $\leftarrow$  bestSolution
  END IF
  SET tabu move
  penalty  $\leftarrow$   $\infty$ 
END FOR
LOOP UNTIL there is no improvement is obtained

END PROCEDURE

```

Figure 4.12: Stage 2 Tabu Search Algorithm

4.9 Conclusion

The algorithms used in this study to solve the UCTTP at the FCSIT was explored in this chapter. The general framework for solving the UCTTP at the FCSIT, UNIMAS was illustrated, which consisted of two stages. The first stage is utilised to generate a feasible initial solution and the second stage is used to improve the quality of the solution. In stage 1, heuristic ordering (HO), heuristic ordering with a perturbation (HO_P) and tabu search (TS) are utilised to generate a feasible initial solution. In stage 2, variable neighbourhood descent (VND) and variable neighbourhood descent with a perturbation are utilised to

improve the quality of the solution. At the end of the chapter, tabu search algorithm is introduced as a comparison with the improved VND algorithm.

CHAPTER 5

REAL-WORLD UNIVERSITY COURSE TIMETABLING PROBLEM

5.1 Introduction

In this chapter, the experimental results using a simulator developed based on the real-world scenario at the Faculty of Computer Science and Information Technology (FCSIT), Universiti Malaysia Sarawak (UNIMAS) are presented. The timetable generated using the proposed algorithm is then compared with the existing timetable. This chapter also provides a discussion of the experimental results.

In stage 1, heuristic ordering (HO) is utilised to generate a feasible initial solution by ensuring all the hard constraints are satisfied. HO with a perturbation (HO_P) is proposed for use in stage 1. To the best of the author's knowledge, the existing scientific literature has not attempted to generate a feasible initial solution using HO_P. In stage 1, all the hard constraints must be satisfied to generate a feasible initial solution.

In the second stage, the soft constraints are taken into consideration. Fulfilling the soft constraints is not mandatory. However, the timetable quality will be determined by the fulfilment of the soft constraints identified in this study, so the quality of the timetable can be improved by satisfying the soft constraints as far as possible. In stage 2, VND is proposed to improve the timetable quality. New neighbourhood structures (NS) which have not been attempted in the existing scientific literature are introduced. VND with a perturbation (VND_P) is also proposed in stage 2 to further improve the quality of the timetable generated using the simulator.

In this chapter, the sensitivity analysis of the two-stage heuristic proposed in this study is also presented.

5.2 Dataset Description

Two real semester datasets were gathered from the faculty's administrator/ timetable planner. Table 5.1 shows the data instances used as testbeds. In this study, all the individual courses are referred to as events. The instance of semester 2 is smaller than semester 1 in terms of events, students and course enrolment because third-year students undergo industrial training.

Table 5.1: FCSIT Data Instances for Academic Years 2019/2020

Instance	Events	Rooms	Students	Timeslot requirement	Event enrolment
Semester 1 2019/2020	100	19 (fixed) 4 (shared)	1454	31	5168
Semester 2 2019/2020	77	18 (fixed) 2 (shared)	1040	31	3394

5.3 Stage 1: Heuristic Ordering (HO)

5.3.1 Experimental Results and Discussion

Table 5.2 shows the number of unallocated courses for the initial solutions generated using different HO in stage 1. Each different HO is executed 30 times to find the average and the best results. The results will show which HO is the most efficient in generating a feasible initial solution. For example, if the best result is 0, it means that by using that HO, all the courses managed to be allocated to the timetable by satisfying all the hard constraints.

Table 5.2: Number of Unallocated Courses for Initial Solutions Generated using Different HO in Stage 1. $N = 30$ runs.

HO	Instance			
	Semester 1 (31 timeslots)		Semester 2 (31 timeslots)	
	Best	Average	Best	Average
LD Ascending	13	15.53	6	8.20
LD Descending	4	5.53	0 *	0.63
LE Ascending	16	17.63	7	7.00
LE Descending	5	6.57	2	2.00
(LD + LE) Ascending	14	16.33	5	7.47
(LD + LE) Descending	0 *	4.20	0 *	1.00
Random	9	9.00	5	5.00

As shown in Table 5.2, both LD and (LD + LE) in descending order managed to find feasible solutions for semester 2's instance using 31 timeslots. However, only (LD + LE) in descending order managed to do so for semester 1's instance using the same number of timeslots. This was because the instance of semester 1 was larger than semester 2 in terms of events, students and course enrolment. There were 77 events (282 lecture hours) in semester 2, compared to 100 events (341 lecture hours) in semester 1. Furthermore, the FCSIT has limited timeslots, since Wednesday and Friday afternoons are blocked. As the size of the data instance grows, this makes allocating lecture hours a challenging task in terms of finding feasibility. In comparison, the solution generated by existing timetabling software required 48 timeslots (8.00am – 9.00pm) to achieve feasibility. This consumes extra resources, such as electricity costs.

5.4 Stage 1: HO_P

5.4.1 Experimental Results and Discussion

Table 5.3 shows the comparison of the number of unallocated courses using HO and HO_P in stage 1 for semester 1's instance. Meanwhile, Table 5.4 shows the comparison of the number of unallocated courses using HO and HO_P in stage 1 for semester 2's instance. As shown in Tables 5.3 and 5.4, HO_P managed to produce a better average result than HO using 31 timeslots over 30 runs. By applying a perturbation to HO, the current initial solution was shaken before HO was applied again for the unallocated events. The procedure was terminated after it reached a defined number of iterations. The p values (< 0.05) from the t -tests led to the rejection of the null hypothesis $H_0: \mu_{HO} = \mu_{HO_P}$ and revealed a statistical difference between the mean of HO and HO_P . This demonstrated that HO_P outperformed HO in generating a feasible initial solution in stage 1, and the improvement was significant.

Table 5.3: Comparison of the Number of Unallocated Courses using HO and HO_P in Stage 1 for Semester 1's instance. $N = 30$ runs.

HO	Without perturbation, <i>HO</i>		With perturbation, <i>HO_P</i>		p value (t -test)
	Best	Average	Best	Average	
LD Descending	4	5.53	0*	1.67	0.000
LE Descending	5	6.57	1	2.27	0.000
(LD + LE) Descending	0*	4.20	0*	2.57	0.000

Table 5.4: Comparison of Number of Unallocated Courses using HO and HO_P in Stage 1 for Semester 2's instance. $N = 30$ runs.

HO	Without perturbation, <i>HO</i>		With perturbation, <i>HO_P</i>		<i>p</i> value (<i>t</i> -test)
	Best	Average	Best	Average	
LD Descending	0*	0.63	0*	0.40	0.006
LE Descending	2	2.00	0*	0.00	0.000
(LD + LE) Descending	0*	1.00	0*	0.00	0.000

5.5 Stage 2: Variable Neighbourhood Descent (VND)

5.5.1 Experimental Results and Discussion

Figure 5.1 shows the comparison between the number of soft constraint violations that occurred using VND_Literature (NS1, NS2, NS5 and NS6) and VND_Proposed (NS1, NS2, NS3, NS4, NS5, NS6, NS7 and NS8) for semester 1's instance. NS3, NS4, NS7 and NS8 are the new NS proposed in this study, whereas NS1, NS2, NS5 and NS6 are the NS found in the scientific literature (Aladag et al., 2009). The average soft constraint violations using VND_Literature and VND_Proposed were 1566.83 and 872.77, respectively. The soft constraint violations are calculated based on the objective function defined in Chapter 3. The lower value of soft constraint violations means the solution generated is of better quality. If the soft constraint violation is not 0, it means some courses are not managed to be split over the minimum number of working days, some students have more than five continuous lecture hours and some students have isolated courses with one lecture hour. By applying the new NS in VND_Proposed, it managed to reduce the soft constraint violations by 44.30% for semester 1's instance. VND_Proposed also managed to produce an improved best soft constraint violation of 425, compared to the VND_Literature result of 1007. The *p* value ($0.000 < 0.05$) from the *t*-test led to the rejection of the null hypothesis $H_0: \mu_{VND_Literature} =$

$\mu_{VND_Proposed}$ and revealed a statistical difference between the means of VND_Literature and VND_Proposed using the new NS. This demonstrated that VND_Proposed outperformed VND_Literature in generating a quality solution in stage 2. The improvement was significant as it involved the introduction of new NS: NS3, NS4, NS7 and NS8.

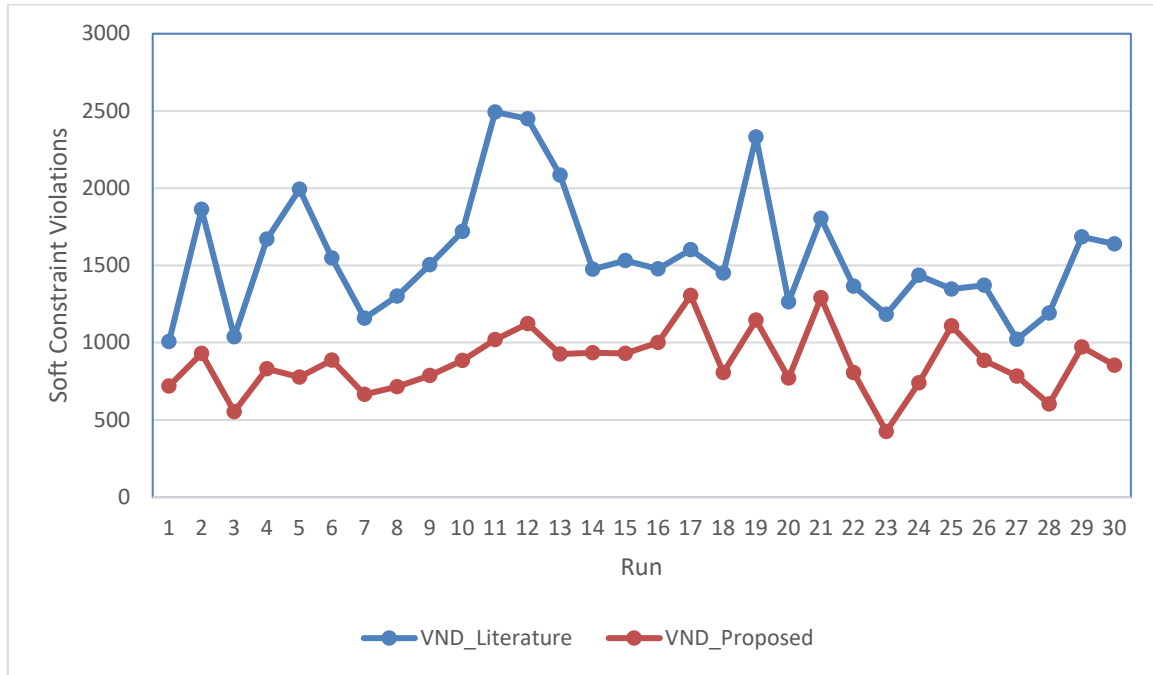


Figure 5.1: Comparison Between the Number of Soft Constraint Violations that occurred using VND_Literature and VND_Proposed for Semester 1's Instance. $N = 30$ runs.

Figure 5.2 shows the comparison between the number of soft constraint violations that occurred using VND_Literature (NS1, NS2, NS5 and NS6) and VND_Proposed (NS1, NS2, NS3, NS4, NS5, NS6, NS7 and NS8) for semester 2's instance. The average soft constraint violations using VND_Literature and VND_Proposed were 704.57 and 416.60, respectively. By applying the new NS in VND_Proposed, it managed to reduce the soft constraint violations by 40.87% for semester 2's instance. VND_Proposed also managed to produce an improved best soft constraint violation of 80, compared to the VND_Literature

result of 136. The p value ($0.000 < 0.05$) from the t -test led to the rejection of the null hypothesis $H_0: \mu_{VND_Literature} = \mu_{VND_Proposed}$ and revealed a statistical difference between the means of VND_Literature and VND_Proposed with the new NS. This demonstrated that VND_Proposed outperformed VND_Literature in generating a quality solution in stage 2. The improvement is significant as it involved introducing new NS: NS3, NS4, NS7 and NS8. This is because having more NS allows more of the solution search space to be explored.

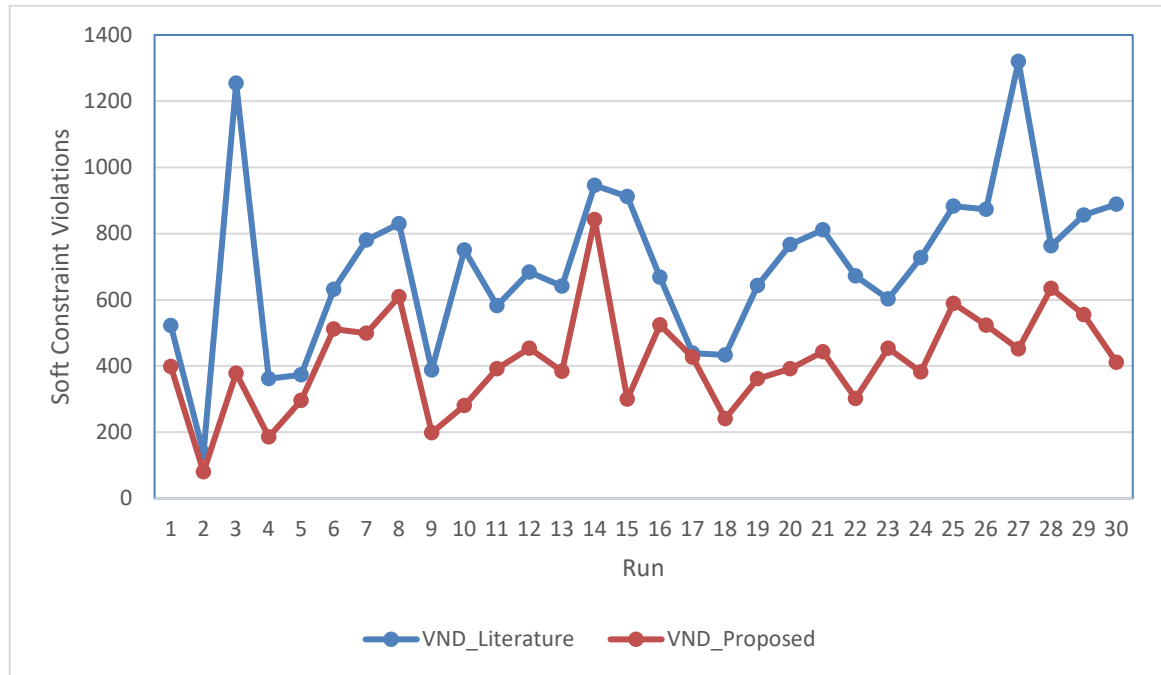


Figure 5.2: Comparison Between the Number of Soft Constraint Violations that occurred using VND_Literature and the VND_Proposed for Semester 2's Instance. $N = 30$ runs.

Table 5.5 shows the comparison between the number of soft constraint violations that occurred using VND_Proposed and the existing FCSIT solution. Based on Table 5.5, the solution generated using VND_Proposed outperformed the existing solution in terms of quality. Furthermore, the existing solution requires 48 timeslots, whereas VND_Proposed

would only require 31 timeslots to be feasible. This would aid the FCSIT in effectively managing its resources.

Table 5.5: Comparison Between the Number of Soft Constraint Violations that occurred using VND_Proposed and the Existing FCSIT Solution. $N = 30$ runs.

Solution	Timeslots	Soft Constraint Violations			
		Semester 1's Instance		Semester 2's Instance	
		Best	Average	Best	Average
Existing FCSIT Solution	48 (8 am to 9 pm)	881		1188	
VND_Proposed	31 (8 am to 5 pm)	425	872.77	80	416.6

In a further analysis, the proposed VND was executed 30 times again for both instances (semesters 1 and 2) to determine which new NS were more efficient, NS3 and NS7 or NS4 and NS8. NS3 and NS7 moved two and one lecture hours, respectively, whereas, NS4 and NS8 swapped two and one lecture hours, respectively. Figure 5.3 shows the comparison of the number of soft constraint violations that occurred using VND_Literature, VND_1|2|3|5|6|7 and VND_1|2|4|5|6|8 for semester 1's instance. The average soft constraint violations using VND_Literature, VND_1|2|3|5|6|7 and VND_1|2|4|5|6|8 were 1566.83, 888.27 and 1261.80, respectively. By applying the new NS (NS3 and NS7) in VND_1|2|3|5|6|7 and the new NS (NS4 and NS8) in VND_1|2|4|5|6|8, they managed to reduce the soft constraint violations by 43.31% and 19.47%, respectively, compared to VND_Literature for semester 1's instance.

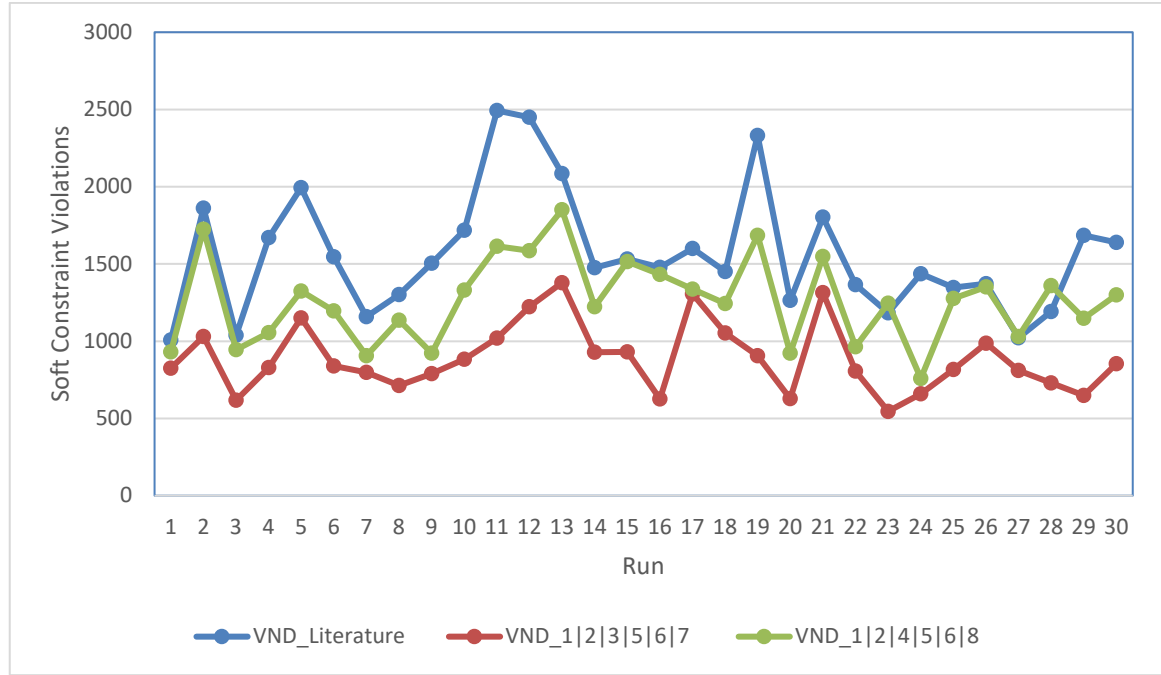


Figure 5.3: Comparison of the Number of Soft Constraint Violations that Occurred Using VND_Literature, VND_1|2|3|5|6|7 and VND_1|2|4|5|6|8 for Semester 1's Instance. $N = 30$ runs.

Figure 5.4 shows the comparison of the number of soft constraint violations that occurred using VND_Literature, VND_1|2|3|5|6|7 and VND_1|2|4|5|6|8 for semester 2's instance. The average soft constraint violations using VND_Literature, VND_1|2|3|5|6|7 and VND_1|2|4|5|6|8 were 704.57, 448.77 and 575.20, respectively. By applying the new NS (NS3 and NS7) in VND_1|2|3|5|6|7 and the new NS (NS4 and NS8) in VND_1|2|4|5|6|8, they managed to reduce the soft constraint violations by 36.31% and 18.36%, respectively, compared to VND_Literature for semester 2's instance. These results show that the new NS of NS3 and NS7 were more efficient than the new NS of NS4 and NS8 in improving the quality of the solution for both instances (semesters 1 and 2). This is because many low-capacity venues and timeslots were still available after a feasible initial solution had been generated in stage 1. For example, semester 1's instance with 341 lecture hours required 341 timeslots and venues, whereas 713 timeslots and venues were available for that semester.

Thus, a simple move managed to provide a hugely significant result when it was applied to the proposed algorithm.

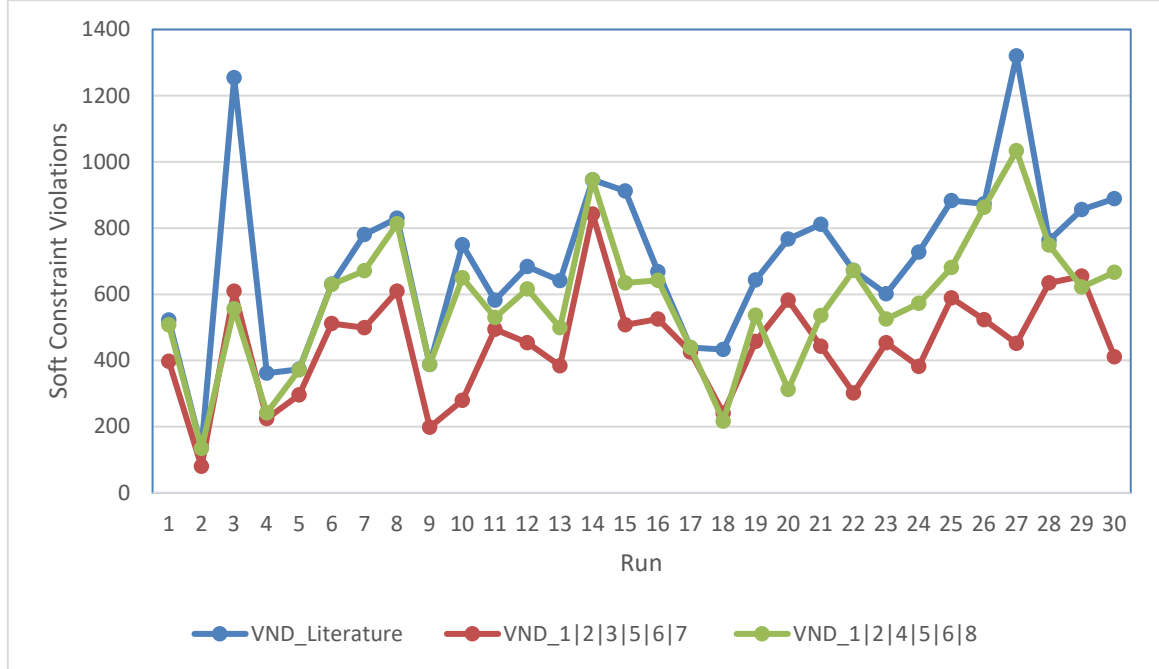


Figure 5.4: Comparison of the Number of Soft Constraint Violations that Occurred Using VND_Literature, VND_1|2|3|5|6|7 and VND_1|2|4|5|6|8 for Semester 2's Instance. $N = 30$ runs.

Paired samples t -tests were used to demonstrate the significance of the improvement using the proposed algorithm. Table 5.6 summarises the soft constraint violations that occurred using the new NS (NS3 and NS7) in stage 2 for both instances (semesters 1 and 2) and the paired samples t -test results. As shown in Table 5.6, when the proposed VND_1|2|3|5|6|7 algorithm was executed 30 times, it produced a higher-quality solution than the VND_Literature. By applying the new NS (NS3 and NS7) together with VND_Literature, the exploration of the search space increased and a better-quality solution was produced. The p value ($0.000 < 0.05$) from the t -test for both instances led to the rejection of the null hypothesis $H_0: \mu_{VND_Literature} = \mu_{VND_1|2|3|5|6|7}$ and revealed a statistical difference between the means of VND_Literature and VND_1|2|3|5|6|7. This demonstrated that VND_1|2|3|5|6|7 outperformed

VND_Literature in generating a quality solution in stage 2. The improvement is significant as it involved introducing new NS: NS3 and NS7.

Table 5.6: Summary of Soft Constraint Violations for the New NS (NS3 and NS7) in Stage 2 for both instances (semesters 1 and 2) and their Paired Samples *t*-test Results. $N = 30$ runs.

Instance	VND_Literature		VND_1 2 3 5 6 7		<i>p</i> value (<i>t</i> -test)
	Best	Average	Best	Average	
Semester 1 2019/2020	1007	1566.83	546	888.27	0.000
Semester 2 2019/2020	136	704.57	80	448.77	0.000

Table 5.7 shows the summary of the soft constraint violations that occurred using the new NS (NS4 and NS8) in stage 2 for both instances (semesters 1 and 2) and the paired samples *t*-test results. As shown in Table 5.7, when the proposed VND_1|2|4|5|6|8 algorithm was executed 30 times, it produced a higher-quality solution than VND_Literature. By applying the new NS (NS4 and NS8) together with VND_Literature, the exploration of the search space increased and a better-quality solution was produced. The *p* value ($0.000 < 0.05$) from the *t*-test for both instances led to the rejection of the null hypothesis $H_0: \mu_{VND_Literature} = \mu_{VND_1|2|4|5|6|8}$ and revealed a statistical difference between the means of VND_Literature and VND_1|2|4|5|6|8. This demonstrated that VND_1|2|4|5|6|8 outperformed VND_Literature in generating a quality solution in stage 2. The improvement is significant as it involved introducing new NS: NS4 and NS8.

Table 5.7: Summary of Soft Constraint Violations for the New NS (NS4 and NS8) in Stage 2 for Both Instances (semesters 1 and 2) and their Paired Samples *t*-test Results. $N=30$ runs.

Instance	VND_Literature		VND_1 2 4 5 6 8		<i>p</i> value (<i>t</i> -test)
	Best	Average	Best	Average	
Semester 1 2019/2020	1007	1566.83	759	1261.80	0.000
Semester 2 2019/2020	136	704.57	134	575.20	0.000

5.6 Stage 2: Variable Neighbourhood Descent (VND) with a Perturbation

5.6.1 Experimental Results and Discussion

To help the current solution to escape from local optima, a perturbation was applied to shake the current solution before the local search continued until a specified stopping condition. Figure 5.5 shows the comparison of the soft constraint violations that occurred using VND_Proposed (NS1, NS2, NS3, NS4, NS5, NS6, NS7 and NS8) and VND_Proposed with a perturbation (VND_Proposed_P) in stage 2 for semester 1's instance. The average soft constraint violations using VND_Proposed and VND_Proposed_P were 872.77 and 625.60, respectively. By applying a perturbation to VND_Proposed, it managed to reduce the soft constraint violations by 28.30% for semester 1's instance. The perturbation procedure also managed to produce an improved best soft constraint violations of 221, compared to the VND_Proposed result of 425.

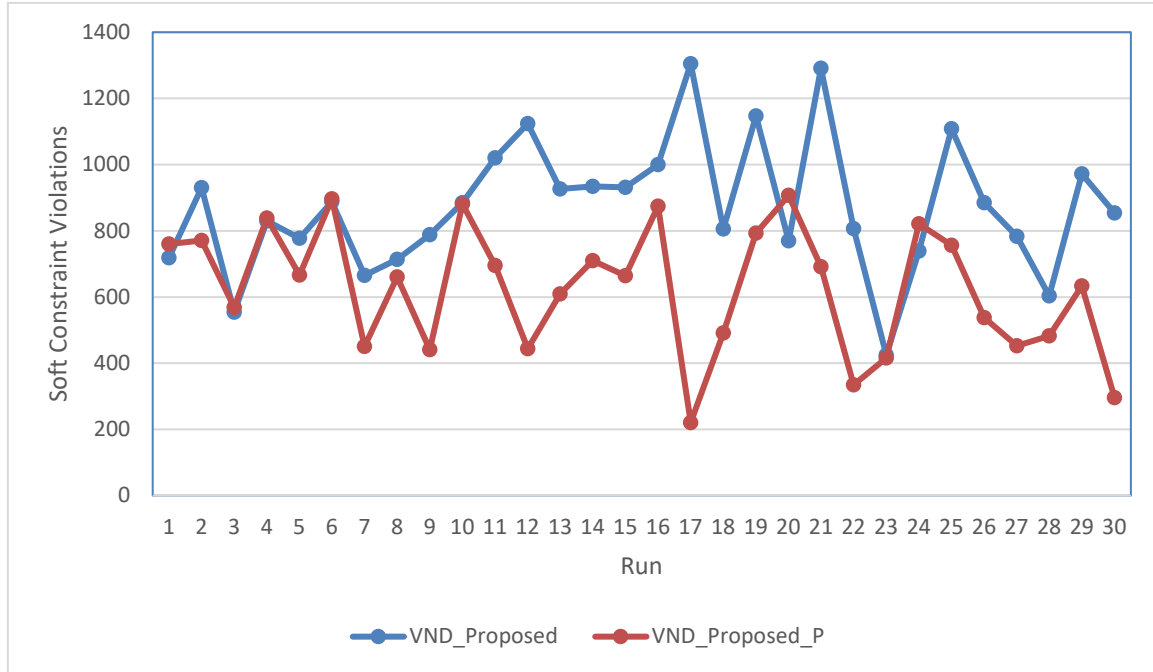


Figure 5.5: Comparison of the Soft Constraint Violations that Occurred Using VND_Proposed and VND_Proposed_P for Semester 1's Instance. $N = 30$ runs.

Figure 5.6 shows the comparison of the soft constraint violations that occurred using VND_Proposed (NS1, NS2, NS3, NS4, NS5, NS6, NS7 and NS8) and VND_Proposed with a perturbation (VND_Proposed_P) in stage 2 for semester 2's instance. The average soft constraint violations using VND_Proposed and VND_Proposed_P were 416.60 and 348.30, respectively. By applying a perturbation to VND_Proposed, it managed to reduce the soft constraint violations by 16.40% for semester 2's instance. The perturbation procedure also managed to produce an improved best soft constraint violations of 18, compared to the VND_Proposed result of 80.

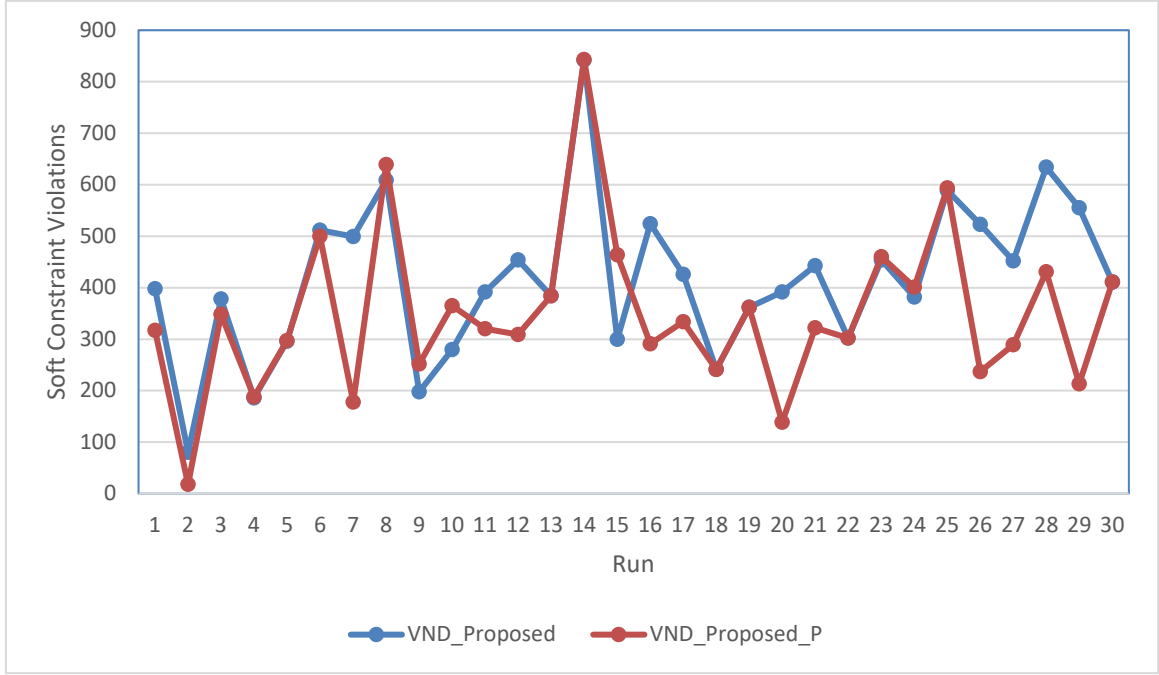


Figure 5.6: Comparison of the Soft Constraint Violations that Occurred Using VND_Proposed and VND_Proposed_P for Semester 2's Instance. $N = 30$ runs.

Paired samples t -tests were used to demonstrate the significance of the improvement obtained using the proposed algorithm. Table 5.8 shows the comparison of the soft constraint violations that occurred using VND_Proposed and VND_Proposed_P in stage 2 for both instances (semesters 1 and 2) and the paired samples t -test results. VND_Proposed_P managed to produce a better-quality solution after 30 runs of the algorithm. By applying a perturbation to VND_Proposed, the current solution was shaken before the VND local search was applied again. This managed to produce a better-quality solution by escaping local optima. The procedure was terminated after no improvement was obtained from the process. The p value (< 0.05) from the t -test for both instances led to the rejection of the null hypothesis H_0 : $\mu_{VND_Proposed} = \mu_{VND_Proposed_P}$ and revealed a statistical difference between the means of VND_Proposed and VND_Proposed_P. This demonstrated that VND_Proposed_P

outperformed VND_Proposed in generating a quality solution in stage 2, and the improvement was significant.

Table 5.8: Comparison of Soft Constraint Violations between VND_Proposed and VND_Proposed_P in Stage 2 for Both Instances (Semesters 1 and 2) and their Paired Samples *t*-test Results. $N = 30$ runs.

Instance	VND_Proposed		VND_Proposed_P		<i>p</i> value (<i>t</i> -test)
	Best	Average	Best	Average	
Semester 1 2019/2020	425	872.77	221	625.60	0.000
Semester 2 2019/2020	80	416.60	18	348.30	0.005

Table 5.9 shows the comparison of the soft constraint violations that occurred when using VND_Proposed, VND_Proposed_P and the existing FCSIT solution. As illustrated in Table 5.9, in terms of the average soft constraint violations, applying VND_Proposed and VND_Proposed_P managed to reduce the soft constraint violations by 0.93% and 28.99%, respectively, compared to the existing solution, for semester 1's instance. Meanwhile, in terms of the average soft constraint violations, using VND_Proposed and VND_Proposed_P managed to reduce the soft constraint violations by 64.93% and 70.68%, respectively, compared to the existing solution, for semester 2's instance. The semester 2 result is more promising than that of semester 1 because the instance of semester 1 is larger than that of semester 2 in terms of events, students and course enrolment.

Table 5.9: Comparison of Soft Constraint Violations between VND_Proposed, VND_Proposed_P and Existing FCSIT Solution. $N = 30$ runs.

Solution	Timeslots	Soft Constraint Violations			
		Semester 1's Instance		Semester 2's Instance	
		Best	Average	Best	Average
Existing FCSIT Solution	48 (8 am to 9 pm)	881		1188	
VND_Proposed	31(8 am to 5 pm)	425	872.77	80	416.6
VND_Proposed_P	31(8 am to 5 pm)	221	625.60	18	348.3

5.7 Sensitivity Analysis

In this section, the sensitivity analysis of the algorithm proposed in this study is presented. The initial solution generated when the problem instance features had been changed is analysed.

The following problem instance features were changed:

- i. Number of events
- ii. Number of available timeslots and large venues
- iii. Number of students

Semester 1's instance was chosen for the sensitivity analysis because the instance of semester 1 is larger than that of semester 2 in terms of events, students and course enrolment.

5.7.1 Number of Events

To analyse the implications of the number of events on timetable feasibility, two events were added to the database for semester 1's instance. Table 5.10 shows the feature of

two extra events added to the database with their corresponding enrolments. Table 5.11 shows the comparison of the feature for semester 1's instance before and after the addition of the two extra events. From the 23 venues available for allocation, nine venues have a maximum capacity of 40 students. For the solution generated using the simulator, most venues with a maximum capacity of 40 students were not being fully utilised. Therefore, the two events added for the sensitivity analysis were assigned to enrolments of less than 40. The students enrolled on the extra events were current students selected randomly from the dataset.

Table 5.10: Feature of Two Extra Events Added to the Database for Semester 1's instance for Sensitivity Analysis

Event	1	2
Number of students	37	38
Number of lecture hours	4	4

Table 5.11: Comparison of the Feature for Semester 1's Instance Before and After the Extra Two Events are Added

Features	Instances	
	Semester 1 – original	Semester 1- after adding 2 extra events
Number of students	1454	1454
Number of enrolments	5168	5243
Number of events	100	102
Number of lecture hours	341	349

After adding the extra two events, the simulator was run again to find the feasibility of the solution. Table 5.12 shows the number of unallocated courses for the initial solutions

generated using different HO in stage 1 after two extra events had been added. As shown in Table 5.12, only (LD + LE) in descending order for semester 1's instance managed to find a feasible solution using 31 timeslots. However, when two extra events were added to semester 1's instance, none of the HO managed to find a feasible solution. The best solution was obtained using (LD + LE) in descending order with three unallocated events. Since HO alone did not manage to provide a feasible initial solution for semester 1's instance using 31 timeslots after the additional events, the proposed algorithm for stage 1 was improved further. A perturbation was applied to the HO to find a feasible solution using 31 timeslots before other techniques were applied, such as adding extra timeslots or extra-large venues.

Table 5.12: Number of Unallocated Courses for Initial Solutions Generated Using Different HO in Stage 1 After Two Extra Events are Added to Semester 1's Instance. $N = 30$ runs.

HO	Semester 1 (31 timeslots) – original		Semester 1 (31 timeslots) – Added 2 events	
	Best	Average	Best	Average
LD Descending	4	5.53	5	7.63
LE Descending	5	6.57	7	7.47
(LD + LE) Descending	0*	4.20	3	4.87
Random	9	9.00	10	10.00

Table 5.13 shows the comparison of the number of unallocated courses for the initial solutions generated using HO and HO_P in stage 1 for semester 1's instance after the extra two events had been added. Meanwhile, Table 5.14 shows the comparison of the number of unallocated courses for semester 1's instance (original) and semester 1's instance (extra two events) in stage 1 for HO and HO_P. As shown in Table 5.13, HO_P managed to produce a

better average result than HO using 31 timeslots over 30 runs. By applying a perturbation to HO, the current initial solution was shaken before HO was applied again for the unassigned events. The procedure was terminated after it reached a defined number of iterations. The p values (< 0.05) from the t -tests led to the rejection of the null hypothesis $H_0: \mu_{HO} = \mu_{HO_P}$ and revealed a statistical difference between the means of HO and HO_P. This demonstrated that HO_P outperformed HO in generating a feasible initial solution in stage 1, and the improvement was significant. Even though the addition of more events increased the complexity of finding a feasible initial solution, it could still be achieved using the proposed algorithm. This result depicts the flexibility of the simulator in solving the UCTTP at the FCSIT.

Table 5.13: Comparison of Number of Unallocated Courses for Initial Solutions Generated Using HO and HO_P in Stage 1 for Semester 1's Instance After the Extra Two Events are Added. $N = 30$ Runs.

HO	Without perturbation, <i>HO</i>		With perturbation, <i>HO_P</i>		p value (t -test)
	Best	Average	Best	Average	
LD Descending	5	7.63	2	2.90	0.000
LE Descending	7	7.47	0	2.57	0.000
(LD + LE) Descending	3	4.87	1	3.17	0.000

Table 5.14: Comparison of Number of Unallocated Courses for Semester 1's Instance (Original) and Semester 1's Instance (Extra Two Events) in Stage 1 for HO and HO_P. $N = 30$ runs.

HO	Semester 1 (31 timeslots) – original				Semester 1 (31 timeslots) – Added 2 events			
	Without perturbation, <i>HO</i>		With perturbation, <i>HO_P</i>		Without perturbation, <i>HO</i>		With perturbation, <i>HO_P</i>	
	Best	Average	Best	Average	Best	Average	Best	Average
LD Descending	4	5.53	0	1.67	5	7.63	2	2.90
LE Descending	5	6.57	1	2.27	7	7.47	0	2.57
(LD + LE) Descending	0	4.20	0	2.57	3	4.87	1	3.17

In a further sensitivity analysis, five events were added to the original database for semester 1's instance. Table 5.15 shows the feature of five extra events added to the original database with their corresponding enrolments. Table 5.16 shows the comparison of the feature for semester 1's instance before and after the addition of five extra events. The five events were added to increase the number of events in the real-world dataset by 5%. A larger number of enrolments was selected because unallocated events usually have a large number of enrolments that require a shared venue. The students enrolled in the extra events were current students selected randomly from the real-world dataset.

Table 5.15: Feature of Five Extra Events Added to The Database for Semester 1's Instance for Sensitivity Analysis

Event	1	2	3	4	5
Number of students	37	38	50	32	143
Number of lecture hours	4	4	4	4	4

Table 5.16: Comparison of the Feature for Semester 1's Instance Before and After the Extra Five Events are Added

Features	Instances	
	Semester 1 – original	Semester 1- after added 5 extra events
Number of students	1454	1454
Number of enrolments	5168	5468
Number of events	100	105
Number of lecture hours	341	361

After adding the five extra events, the simulator was run again to find the feasibility of the solution. Table 5.17 shows the number of unallocated courses for the initial solutions generated using different HO in stage 1 after five extra events had been added to semester 1's instance. As shown in Table 5.17, only (LD + LE) in descending order for semester 1's instance managed to find a feasible solution using 31 timeslots. However, when the five extra events were added to semester 1's instance, none of the HO managed to find a feasible solution. The best solution was obtained using (LD + LE) in descending order with three unallocated events. Since HO alone did not manage to provide a feasible initial solution for semester 1's instance using 31 timeslots after the additional events, the proposed algorithm for stage 1 was improved further. A perturbation was applied to the HO to find a feasible solution using 31 timeslots before other techniques were applied, such as adding extra timeslots or extra-large venues.

Table 5.17: Number of Unallocated Courses for Initial Solutions Generated Using Different HO in Stage 1 After Five Extra Events are Added to Semester 1's Instance. $N = 30$ runs.

HO	Semester 1 (31 timeslots) – original		Semester 1 (31 timeslots) – Added 5 events	
	Best	Average	Best	Average
LD Descending	4	5.53	7	9.13
LE Descending	5	6.57	6	8.20
(LD + LE) Descending	0	4.20	3	6.07
Random	9	9.00	12	12.00

Table 5.18 shows the comparison of the number of unallocated courses for the initial solutions generated using HO and HO_P in stage 1 for semester 1's instance after the addition of five extra events. As shown in Table 5.18, HO_P managed to produce a better average result than HO using 31 timeslots over 30 runs. By applying a perturbation to HO, the current initial solution was shaken before HO was applied again for the unassigned events. The procedure was terminated after a defined number of iterations. The p values (< 0.05) from the t -tests led to the rejection of the null hypothesis $H_0: \mu_{HO} = \mu_{HO_P}$ and revealed a statistical difference between the means of HO and HO_P. This demonstrated that HO_P outperformed HO in generating a feasible initial solution in stage 1, and the improvement was significant. Table 5.19 shows the comparison of the number of unallocated courses for semester 1's instance (original) and semester 1's instance (extra five events) in stage 1 for HO and HO_P. As illustrated in Table 5.19, after five extra events had been added to semester 1's original instance, a feasible initial solution could not be generated using either HO or HO_P. Thus, other techniques could be used to generate a feasible initial solution, such as adding more timeslots or using alternative algorithms.

Table 5.18: Comparison of Number of Unallocated Courses for Initial Solutions Generated Using HO and HO_P in Stage 1 for Semester 1's Instance After the Extra Five Events are Added. $N = 30$ runs.

HO	Without perturbation, <i>HO</i>		With perturbation, <i>HO_P</i>		<i>p</i> value (<i>t</i> -test)
	Best	Average	Best	Average	
LD Descending	7	9.13	1	2.63	0.000
LE Descending	6	8.20	3	4.40	0.000
(LD + LE) Descending	3	6.07	2	4.17	0.000

Table 5.19: Comparison of Number of Unallocated Courses for Semester 1's Instance (Original) and Semester 1's Instance (Extra Five Events) in Stage 1 for HO and HO_P. $N = 30$ runs.

HO	Semester 1 (31 timeslots) – original				Semester 1 (31 timeslots) – Added 5 events			
	Without perturbation, <i>HO</i>		With perturbation, <i>HO_P</i>		Without perturbation, <i>HO</i>		With perturbation, <i>HO_P</i>	
	Best	Average	Best	Average	Best	Average	Best	Average
LD Descending	4	5.53	0	1.67	7	9.13	1	2.63
LE Descending	5	6.57	1	2.27	6	8.20	3	4.40
(LD + LE) Descending	0	4.20	0	2.57	3	6.07	2	4.17

5.7.2 Timetable Availability and Large Venue Resources Availability

The simulator used to generate the timetable for semester 1's instance managed to generate a feasible initial solution in stage 1 using the proposed algorithm. However, when five additional events were added with an extra 20 lecture hours per week, there was one unallocated course if HO (LD Descending) with a perturbation was used to generate the initial solution in stage 1. Another approach has been used to generate the initial solution

which is Tabu Search (TS). Table 5.20 shows the comparison of the number of unallocated courses for semester 1's instance (extra five events) in stage 1 for HO and TS with 31 timeslots. As depicted in Table 5.20, TS did not manage to find a feasible initial solution after five extra events had been added to semester 1's original instance.

Table 5.20: Comparison of Number of Unallocated Courses for Semester 1's Instance (Extra Five Events) in Stage 1 for HO and TS with 31 Timeslots. $N = 30$ runs.

HO	Semester 1 (31 timeslots) – Added 5 events (HO)		Semester 1 (31 timeslots) – Added 5 events (TS)	
	Best	Average	Best	Average
LD Descending	7	9.13	6	8.37
LE Descending	6	8.20	5	7.43
(LD + LE) Descending	3	6.07	4	6.53

As a result, increasing the number of available timeslots is one of the tactics that can be used because this also increases the number of shared venues available. Two options can be applied. The first is to extend the extra timeslots after five pm and the second is to allocate the unallocated courses to reserved timeslots.

Option 1: Extend the timetable by one hour to 6 pm on Monday, Tuesday and Thursday.

A total of three timeslots were added to the original 31 timeslots of the proposed algorithm. After the addition of these timeslots, the simulator was run again to get the new results. Figure 5.7 shows the original timeslot availability and the timeslot availability after the addition of three timeslots from 5 pm to 6 pm on Monday, Tuesday and Thursday.

Original Timeslots Availability

Day\Time	0800-0900	0900-1000	1000-1100	1100-1200	1200-1300	1300-1400	1400-1500	1500-1600	1600-1700
Monday	01	02	03	04	05	06	07	08	09
Tuesday	10	11	12	13	14	15	16	17	18
Wednesday									
Thursday	19	20	21	22	23	24	25	26	27
Friday	28	29	30	31					

Timeslots Availability after three hours timeslots are added

Day\Time	0800-0900	0900-1000	1000-1100	1100-1200	1200-1300	1300-1400	1400-1500	1500-1600	1600-1700	1700-1800
Monday	01	02	03	04	05	06	07	08	09	10*
Tuesday	11	12	13	14	15	16	17	18	19	20*
Wednesday										
Thursday	21	22	23	24	25	26	27	28	29	30*
Friday	31	32	33	34						

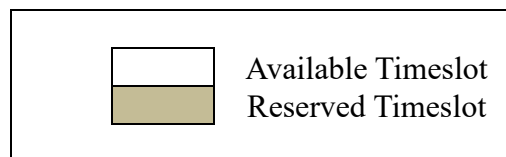


Figure 5.7: Original Timeslots Availability and Timeslots Availability After Three Hours Timeslots are Added

Table 5.21 shows the comparison of the number of unallocated courses for semester 1's instance (extra five events) in stage 1 for HO when the number of available timeslots was increased from 31 to 34 for the sensitivity analysis. When the number of timeslots was increased, the number of shared large-capacity venues also increased. As illustrated in Table 5.21, after

the addition of these timeslots shown in Figure 5.7, the results show that a feasible initial solution could not be found for any HO. However, when three timeslots were added, the average number of unallocated courses managed to be improved for all the HO. Table 5.22 shows the comparison of the number of unallocated courses for semester 1's instance (extra five events) in stage 1 for HO_P when the number of available timeslots was increased from 31 to 34 for the sensitivity analysis. As depicted in Table 5.22, after the addition of these timeslots shown in Figure 5.7, HO_P managed to find a feasible initial solution using LD Descending and (LD+LE) Descending.

Table 5.21: Comparison of Number of Unallocated Courses for Semester 1's Instance (Extra Five Events) in Stage 1 for HO When Available Timeslots are Increased from 31 Timeslots to 34 Timeslots. $N = 30$ runs.

HO	Semester 1 (31 timeslots) – Added 5 events		Semester 1 (34 timeslots) – Added 5 events	
	Best	Average	Best	Average
LD Descending	7	9.13	1	2.00
LE Descending	6	8.20	4	5.20
(LD + LE) Descending	3	6.07	1	2.83

Table 5.22: Comparison of Number of Unallocated Courses for Semester 1's Instance (Extra Five Events) in Stage 1 for HO_P When Available Timeslots are Increased from 31 Timeslots to 34 Timeslots. $N = 30$ runs.

HO	Semester 1 (31 timeslots) – Added 5 events		Semester 1 (34 timeslots) – Added 5 events	
	Best	Average	Best	Average
LD Descending	1	2.63	0	0.37
LE Descending	3	4.40	1	2.30
(LD + LE) Descending	2	4.17	0	0.83

Option 2: Open up reserved timeslots on Wednesday

A total of three reserved timeslots were added to the original 31 timeslots of the proposed algorithm. Figure 5.8 shows the original timeslot availability and the timeslot availability after three timeslots had been added using reserved timeslots.

Original Timeslots Availability

Day\Time	0800-0900	0900-1000	1000-1100	1100-1200	1200-1300	1300-1400	1400-1500	1500-1600	1600-1700
Monday	01	02	03	04	05	06	07	08	09
Tuesday	10	11	12	13	14	15	16	17	18
Wednesday									
Thursday	19	20	21	22	23	24	25	26	27
Friday	28	29	30	31					

After three hours timeslots are added

Day\Time	0800-0900	0900-1000	1000-1100	1100-1200	1200-1300	1300-1400	1400-1500	1500-1600	1600-1700
Monday	01	02	03	04	05	06	07	08	09
Tuesday	10	11	12	13	14	15	16	17	18
Wednesday	19	20	21						
Thursday	22	23	24	25	26	27	28	29	30
Friday	31	32	33	34					

	Available Timeslot
	Reserved Timeslot

Figure 5.8: Original Timeslots Availability and Timeslots Availability After 3 Timeslots are Added Using Reserved Timeslots

5.7.3 Number of Students

To analyse the implications of the number of students on the timetable generated using the simulator, 100 students were added to the original semester 1's instance. Table 5.23 shows the comparison of the feature for semester 1's instance before and after the addition of extra students (100 students) to the original semester 1's instance. One hundred

new students were added to increase the number of students in the real- world dataset by 7%. The enrolments of the new students were made randomly, based on the curriculum plan.

Table 5.23: Comparison of the Feature for Semester 1's Instance Before and After the Extra Students (100 Students) are Added

Features	Instances	
	Semester 1 – original	Semester 1- after adding 100 students
Number of students	1454	1554
Number of enrolments	5168	5468
Number of events	100	100
Number of lecture hours	341	341

After adding the extra 100 students, the simulator was run again to find the feasibility of the solution. Table 5.24 shows the number of unallocated courses for the initial solutions generated using different HO in stage 1 after 100 extra students had been added to semester 1's instance. As shown in Table 5.24, only (LD + LE) in descending order for semester 1's instance managed to find a feasible solution using 31 timeslots. However, when 100 extra students were added to semester 1's instance, none of the HO managed to find a feasible solution. The best solution was using HO (LD + LE) in descending order with three unallocated events. Since HO alone did not manage to provide a feasible initial solution for semester 1's instance using 31 timeslots after the addition of 100 students, the proposed algorithm for stage 1 was improved further. A perturbation was applied to the HO to find a feasible solution using 31 timeslots before other techniques were applied, such as adding extra timeslots or extra-large venues.

Table 5.24: Number of Unallocated Courses for Initial Solutions Generated Using Different HO in Stage 1 After 100 Extra Students are Added to Semester 1's Instance. $N=30$ runs.

HO	Semester 1 (31 timeslots) – original		Semester 1 (31 timeslots) – Added 100 students	
	Best	Average	Best	Average
LD Descending	4	5.53	5	5.93
LE Descending	5	6.57	6	7.23
(LD + LE) Descending	0	4.20	3	5.03
Random	9	9.00	11	11.00

Table 5.25 shows the comparison of the number of unallocated courses for the initial solutions generated using HO and HO_P in stage 1 for semester 1's instance after the addition of 100 extra students. Meanwhile, Table 5.26 shows the comparison of the number of unallocated courses for semester 1's instance (original) and semester 1's instance (extra 100 students) in stage 1 for HO and HO_P. As shown in Table 5.25, HO_P managed to produce a better average result using 31 timeslots compared to HO over 30 runs. HO (LD Descending) managed to generate a feasible initial solution in stage 1 even after the addition of 100 students. This again proves that HO_P outperformed HO. The p values (< 0.05) from the t -tests led to the rejection of the null hypothesis $H_0: \mu_{HO} = \mu_{HO_P}$ and revealed a statistical difference between the means of HO and HO_P. This demonstrated that HO_P outperformed HO in generating a feasible initial solution in stage 1, and the improvement was significant. Even though the addition of more students increased the complexity of finding a feasible initial solution, it could still be achieved using the proposed algorithm. This result depicts the flexibility of the simulator in solving the UCTTP at the FCSIT.

Table 5.25: Comparison of Number of Unallocated Courses for Initial Solutions Generated Using HO and HO_P in Stage 1 for Semester 1's Instance After the Extra 100 Students are Added. $N = 30$ runs.

HO	Without perturbation, <i>HO</i>		With perturbation, <i>HO_P</i>		<i>p</i> value (<i>t</i> -test)
	Best	Average	Best	Average	
LD Descending	5	5.93	0	2.47	0.000
LE Descending	6	7.23	2	3.47	0.000
(LD + LE) Descending	3	5.03	2	4.10	0.000

Table 5.26: Comparison of Number of Unallocated Courses for Semester 1's Instance (Original) and Semester 1's Instance (Extra 100 Students) in Stage 1 for HO and HO_P. $N = 30$ runs.

HO	Semester 1 (31 timeslots) – original				Semester 1 (31 timeslots) – Added 100 students			
	Without perturbation, <i>HO</i>		With perturbation, <i>HO_P</i>		Without perturbation, <i>HO</i>		With perturbation, <i>HO_P</i>	
	Best	Average	Best	Average	Best	Average	Best	Average
LD Descending	4	5.53	0	1.67	5	5.93	0	2.47
LE Descending	5	6.57	1	2.27	6	7.23	2	3.47
(LD + LE) Descending	0	4.20	0	2.57	3	5.03	2	4.10

5.8 Comparison Between VND_Proposed and Tabu Search (TS)

5.8.1 Experimental Results and Discussion

Figure 5.9 shows the comparison between the number of soft constraint violations that occurred using VND_Proposed and Tabu Search for semester 1's instance. The average soft constraint violations using VND_Proposed and Tabu Search were 872.77 and 1583.60, respectively. VND_Proposed also managed to produce an improved best soft constraint violation of 425, compared to the Tabu Search result of 1022. The *p* value ($0.000 < 0.05$) from

the t -test led to the rejection of the null hypothesis $H_0: \mu_{VND_Proposed} = \mu_{TabuSearch}$ and revealed a statistical difference between the means of VND_Proposed and Tabu Search. This demonstrated that VND_Proposed outperformed Tabu Search in generating a quality solution in stage 2. The improvement was significant as it involved the introduction of new NS: NS3, NS4, NS7 and NS8.

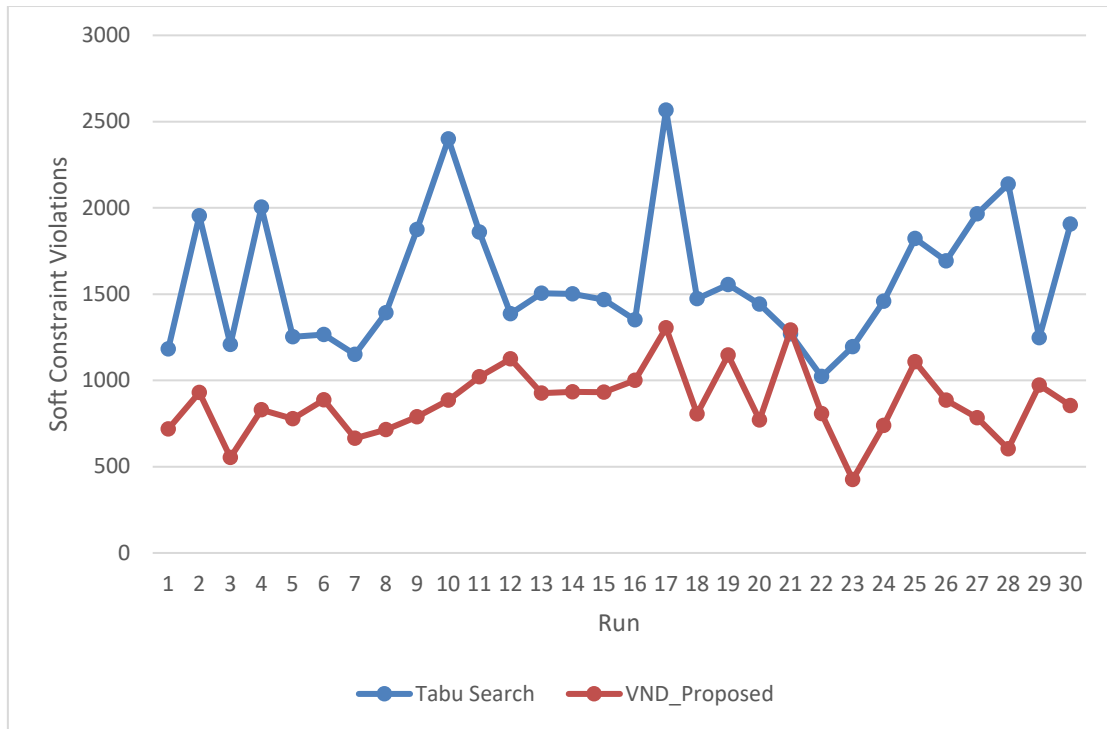


Figure 5.9: Comparison Between the Number of Soft Constraint Violations that occurred using VND_Proposed and Tabu Search for Semester 1's Instance. $N = 30$ runs.

Figure 5.10 shows the comparison between the number of soft constraint violations that occurred using VND_Proposed and Tabu Search for semester 2's instance. The average soft constraint violations using VND_Proposed and Tabu Search were 416.60 and 684.70, respectively. VND_Proposed also managed to produce an improved best soft constraint violation of 80, compared to the Tabu Search result of 245. The p value ($0.000 < 0.05$) from the t -test led to the rejection of the null hypothesis $H_0: \mu_{VND_Proposed} = \mu_{TabuSearch}$ and revealed a

statistical difference between the means of VND_Proposed and Tabu Search. This demonstrated that VND_Proposed outperformed Tabu Search in generating a quality solution in stage 2. The improvement was significant as it involved the introduction of new NS: NS3, NS4, NS7 and NS8.

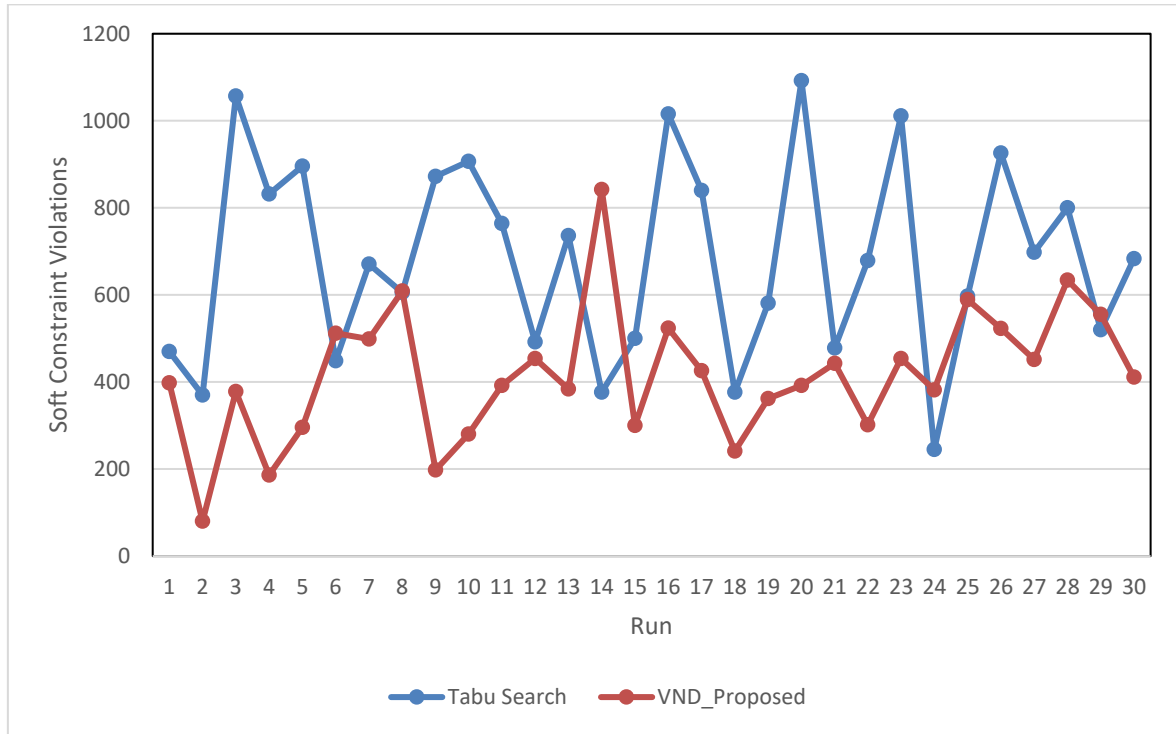


Figure 5.10: Comparison Between the Number of Soft Constraint Violations that occurred using VND_Proposed and Tabu Search for Semester 2's Instance. $N = 30$ runs.

5.9 Conclusion

A two-stage heuristic has been proposed in this study. In stage 1, HO is applied to generate a feasible initial solution. HO (LD and (LD+LE) in descending order) managed to find a feasible solution for semester 2's instance using 31 timeslots but only HO ((LD+LE) in descending order) managed to do so for semester 1's instance using the same number of timeslots. This is because the instance of semester 1 was larger than that of semester 2 in terms of events, students and course enrolment. In comparison, the solution generated by the existing timetabling software required 48 timeslots to achieve feasibility.

To further improve the HO algorithm, a perturbation was applied to the HO to find a feasible solution for semester 1's and semester 2's instances using 31 timeslots. HO and HO_P for both instances (semester 1 and semester 2) were compared in terms of the number of unallocated courses. It was found that HO_P outperformed HO in generating a feasible initial solution in stage 1, and the improvement was significant. This significance was proven using paired samples *t*-tests.

In stage 2, VND with the new NS (NS3, NS4, NS7 and NS8) is proposed in addition to the NS identified in the scientific literature (NS1, NS2, NS5 and NS6) to improve the quality of the feasible initial solution obtained in stage 1. VND (NS1, NS2, NS5 and NS6) and VND (NS1, NS2, NS3, NS4, NS5, NS6, NS7, NS8) were compared in terms of the average soft constraint violations that occurred for both instances (semester 1 and semester 2). It was found that VND (NS1, NS2, NS3, NS4, NS5, NS6, NS7, NS8) outperformed VND (NS1, NS2, NS5 and NS6), and the improvement was significant. This significance was proven using paired samples *t*-tests. It was also found that the quality of the solution generated by using the proposed VND (NS1, NS2, NS3, NS4, NS5, NS6, NS7, NS8) outperformed the existing FCSIT solution.

The sensitivity analysis of the algorithm proposed in this study is presented. The initial solution generated when the problem instance features were changed was analysed. The changes were the number of students, number of events, number of timeslots available for allocation and number of large venues. The result depicts the flexibility of the simulator in solving the UCTTP at the FCSIT. If a feasible initial solution could not be found after the addition of events or students, other techniques could be used to generate a feasible initial solution, such as adding more timeslots or using alternative algorithms.

At the end of the chapter, a comparison is done between VND (NS1, NS2, NS3, NS4, NS5, NS6, NS7, NS8) and TS. It was found that VND (NS1, NS2, NS3, NS4, NS5, NS6, NS7, NS8) outperformed TS with the implementation of the new NS (NS3, NS4, NS7 and NS8).

CHAPTER 6

ITC-2007 (TRACK 2) BENCHMARK DATASET

6.1 Introduction

In this chapter, the experimental results using a simulator developed based on the ITC-2007 (Track 2) benchmark dataset are presented. This chapter also provides a discussion of the experimental results. In this study, ITC-2007 (Track 2) benchmark dataset is chosen because the constraints for this dataset have the most similarities compared to the real dataset.

6.2 Problem Description

The International Timetabling Competition (ITC-2007-Track 2) was established to facilitate comparisons of solution approaches. Many researchers used the ITC-2007 (Track 2) benchmark dataset in their study, for example, Lewis (2010), Ceschia et al. (2012), Nothegger et al. (2012), Nagata (2018) and Goh et al. (2020). In this study, the dataset is used to demonstrate the significant improvement obtained using the proposed algorithm, VND_Proposed, which used new NS (NS7 and NS8). The attributes of the dataset given in the competition differ from those of the real-world dataset. For example, all the ITC-2007 (Track 2) events have only one lecture hour, whereas FCSIT events ranged from two to four lecture hours.

Table 6.1 shows the number of timeslots used in ITC-2007 (Track 2). As shown in Table 6.1, the total number of timeslots is 45, with nine hours available daily from Monday to Friday.

Table 6.1: Number of Timeslots Used in ITC-2007 (Track 2)

Day\Time	8-9	9-10	10-11	11-12	12-13	13-14	14-15	15-16	16-17
Mon	1	2	3	4	5	6	7	8	9
Tue	10	11	12	13	14	15	16	17	18
Wed	19	20	21	22	23	24	25	26	27
Thurs	28	29	30	31	32	33	34	35	36
Fri	37	38	29	40	41	42	43	44	45

The constraints detailed for the ITC-2007 (Track 2) are also divided into hard and soft constraints.

The hard constraints are as listed:

- i. A student is only allowed to attend one event at a time.
- ii. The venue is large enough to accommodate all the students and satisfies all of the event's features.
- iii. In any given timeslot, only one event is allocated in each venue.
- iv. Events are only given timeslots that have already been pre-defined as accessible for specific events.
- v. Events are assigned in accurate order over the week unless otherwise noted.

The soft constraints are as listed:

- i. A student has a class scheduled at the end of the day.
- ii. A student attends more than two classes one after the other.
- iii. On any given day, a student has only one class.

6.3 Dataset Description

The ITC-2007 (Track 2) benchmark dataset has 24 instances. The features of the instances are shown in Table 6.2.

Table 6.2: The Features of the ITC-2007 (Track 2) Benchmark Dataset.

Instances	Students	Events	Rooms	Features
01	500	400	10	10
02	500	400	10	10
03	1000	200	20	10
04	1000	200	20	10
05	300	400	20	20
06	300	400	20	20
07	500	200	20	20
08	500	200	20	20
09	500	400	10	20
10	500	400	10	20
11	1000	200	10	10
12	1000	200	10	10
13	300	400	20	10
14	300	400	20	10
15	500	200	10	20
16	500	200	10	20
17	500	100	10	10
18	500	200	10	10
19	1000	300	10	10
20	1000	400	10	10
21	300	500	20	20
22	500	600	20	20
23	1000	400	20	30
24	1000	400	20	30

6.4 Stage 1: Heuristic Ordering with a Perturbation (HO_P)

6.4.1 Experimental Results and Discussion

Table 6.3 shows the comparison of the number of unallocated courses for the initial solutions generated using HO and HO_P in stage 1 for the ITC-2007 (Track 2) instances. As shown in Table 6.3, HO_P managed to produce a better average result compared to HO for all the ITC-2007 (Track 2) instances over 30 runs for each instance. HO_P also managed to find a feasible initial solution for instances 03, 04, 08, 11, 15, 16 and 20. By applying a perturbation to HO, the current initial solution was shaken before HO was applied again for the unallocated events. The procedure was terminated after it reached a defined number of iterations. The p values (< 0.05) from the t -tests led to the rejection of the null hypothesis $H_0: \mu_{HO} = \mu_{HO_P}$ and revealed a statistical difference between the means of HO and HO_P. This demonstrated that HO_P outperformed HO in generating an initial solution in stage 1, and the improvement was significant.

Table 6.3: Comparison of Number of Unallocated Courses using HO and HO_P in Stage 1 for ITC-2007 (Track 2) Instances. $N = 30$ runs.

Instances	Without perturbation, <i>HO</i>		With perturbation, <i>HO_P</i>		p value (t -test)
	Best	Average	Best	Average	
01	66	75.60	33	47.03	0.000
02	73	85.17	42	58.70	0.000
03	5	8.17	0	0.03	0.000
04	17	19.70	0	1.70	0.000
05	36	46.50	1	6.00	0.000
06	44	49.60	3	7.60	0.000
07	21	25.57	1	10.37	0.000
08	19	23.13	0	1.17	0.000
09	77	84.07	47	57.57	0.000

Table 6.3 continued

10	94	103.43	68	83.27	0.000
11	8	13.57	0	0.80	0.000
12	16	23.67	4	11.17	0.000
13	52	57.07	6	11.13	0.000
14	51	55.97	10	13.93	0.000
15	17	20.03	0	1.47	0.000
16	9	10.77	0	0.67	0.000
17	3	4.27	1	2.73	0.000
18	14	23.37	1	5.60	0.000
19	38	44.00	24	30.13	0.000
20	21	26.23	0	2.83	0.000
21	58	62.77	3	6.17	0.000
22	122	129.93	70	88.40	0.000
23	73	83.13	43	55.90	0.000
24	46	50.20	9	13.50	0.000

6.5 Stage 1: Tabu Search (TS)

6.5.1 Experimental Results and Discussion

Table 6.4 shows the comparison of the number of unallocated courses for the initial solutions generated using HO_P and HO_P + Tabu Search (HO_P_TS) in stage 1 for the ITC-2007 (Track 2) instances. As shown in Table 6.4, HO_P_TS managed to produce a better average result for all the ITC-2007 (Track 2) instances compared to HO_P over 30 runs for each instance. As depicted in Table 6.4, only instance three had a p value ($0.326 > 0.05$), while the rest had p values (< 0.05). The p values (< 0.05) from the t -tests led to the rejection of the null hypothesis $H_0: \mu_{HO_P} = \mu_{HO_P_TS}$ and revealed a statistical difference between the means of HO_P and HO_P_TS. This demonstrated that HO_P_TS

outperformed HO_P in generating an initial solution in stage 1 for the ITC-2007 (Track 2) instances, and the improvement was significant.

Table 6.4: Comparison of Number of Unallocated Courses using HO_P and HO_P_TS in Stage 1 for ITC-2007 (Track 2) Instances. $N = 30$ runs.

Instances	<i>HO_P</i>		<i>HO_P_TS</i>		<i>p</i> value (<i>t</i> -test)
	Best	Average	Best	Average	
01	33	47.03	0	2.27	0.000
02	42	58.70	0	3.73	0.000
03	0	0.03	0	0.00	0.326
04	0	1.70	0	0.00	0.000
05	1	6.00	0	0.03	0.000
06	3	7.60	0	0.07	0.000
07	1	10.37	0	0.00	0.000
08	0	1.17	0	0.00	0.000
09	47	57.57	0	6.07	0.000
10	68	83.27	11	18.10	0.000
11	0	0.80	0	0.00	0.001
12	4	11.17	0	0.00	0.000
13	6	11.13	0	0.07	0.000
14	10	13.93	0	0.10	0.000
15	0	1.47	0	0.00	0.000
16	0	0.67	0	0.00	0.000
17	1	2.73	0	0.00	0.000
18	1	5.60	0	0.00	0.000
19	24	30.13	0	1.77	0.000
20	0	2.83	0	0.00	0.000
21	3	6.17	0	0.13	0.000
22	70	88.40	24	34.00	0.000
23	43	55.90	0	0.50	0.000
24	9	13.50	0	0.03	0.000

6.6 Stage 2: Variable Neighbourhood Descent (VND)

6.6.1 Experimental Results and Discussion

A total of eight NS are used in VND_Proposed for the UNIMAS dataset. NS1, NS2, NS5 and NS6 are the NS used in the literature. NS3, NS4, NS7 and NS8 are the new NS implemented in VND_Proposed. However, only NS5, NS6, NS7 and NS8 are used in the proposed VND algorithm for the ITC-2007 (Track 2) instances. This is because all the ITC-07 (Track 2) events have only one lecture hour, whereas the FCSIT events ranged from two to four lecture hours. Table 6.5 shows the comparison of the soft constraint violations that occurred using VND (NS5 and NS6) and the proposed VND (NS5, NS6, NS7 and NS8) for the ITC-2007 (Track 2) instances. As shown in Table 6.5, the proposed VND (VND_5|6|7|8) with the new NS (NS7 and NS8) managed to produce a better-quality solution, on average, for all the ITC-2007 (Track 2) instances compared to VND_5|6 without the new NS over 30 runs. VND_5|6|7|8 also managed to produce improved best soft constraint violations for all the ITC-2007 (Track 2) instances compared to VND_5|6 over 30 runs of the proposed algorithm. The p values (< 0.05) from the t -tests for all the instances led to the rejection of the null hypothesis $H_0: \mu_{VND_5|6} = \mu_{VND_5|6|7|8}$ and revealed a statistical difference between the means of VND_5|6 and the proposed VND_5|6|7|8. This demonstrated that the proposed VND_5|6|7|8 outperformed VND_5|6 in stage 2 by introducing new NS (NS7 and NS8), and the improvement was significant.

Table 6.5: Comparison of Soft Constraint Violations using VND_5|6 and the Proposed VND_5|6|7|8 for ITC-2007 (Track 2) Instances. $N = 30$ runs.

Instance	VND_5 6		VND_5 6 7 8		p value (t -test)
	Best	Average	Best	Average	
1	2284	2652	2216	2579	0.000
2	2455	2722	2395	2645	0.000
3	1379	2097	1278	1990	0.000
4	1839	2148	1712	1992	0.002
5	1255	1459	1240	1421	0.000
6	1325	1449	1298	1412	0.000
7	1063	1290	1063	1276	0.012
8	968	1208	963	1171	0.000
9	2515	2755	2428	2680	0.000
10	2387	2653	2340	2593	0.000
11	2112	2463	2078	2369	0.000
12	2051	2491	2019	2429	0.000
13	1389	1516	1325	1474	0.000
14	1413	1498	1386	1476	0.000
15	1139	1323	1078	1259	0.000
16	882	1107	756	984	0.000
17	1062	1351	798	1204	0.000
18	1918	2361	1857	2264	0.000
19	2411	2725	2343	2646	0.000
20	2247	2470	1920	2226	0.000
21	1257	1400	1227	1361	0.000
22	3011	3209	2948	3163	0.000
23	4707	5258	4586	5210	0.000
24	2081	2338	2008	2270	0.000

6.7 Stage 2: Variable Neighbourhood Descent (VND) with a Perturbation

6.7.1 Experimental Results and Discussion

To help the current solution to escape from local optima, a perturbation was applied to shake the current solution before the local search continued until a specified stopping condition. Table 6.6 shows the comparison of the soft constraint violations that occurred using the proposed VND_5|6|7|8 and VND_5|6|7|8 with a perturbation (VND_5|6|7|8_P) in stage 2 for ITC-2007 (Track 2). As shown in Table 6.6, the proposed VND_5|6|7|8_P managed to produce a better-quality solution on average over 30 runs. By applying a perturbation to the proposed VND_5|6|7|8, the current solution was shaken before the VND local search was applied again to obtain a better-quality solution by escaping local optima. The procedure was terminated after no improvement was obtained from the process. The p values (< 0.05) from the t -tests for all the instances led to the rejection of the null hypothesis $H_0: \mu_{VND_5|6|7|8} = \mu_{VND_5|6|7|8_P}$ and revealed a statistical difference between the means of VND_5|6|7|8 and VND_5|6|7|8_P. This demonstrated that VND_5|6|7|8_P outperformed VND_5|6|7|8 in generating a quality solution in stage 2, and the improvement was significant.

Table 6.6: Comparison of Soft Constraint Violations between VND_5|6|7|8 and VND_5|6|7|8_P in Stage 2 for ITC-2007 (Track 2). $N = 30$ runs.

Instance	VND_5 6 7 8		VND_5 6 7 8_P		p value (t -test)
	Best	Average	Best	Average	
1	2216	2579	2186	2500	0.000
2	2395	2645	2357	2582	0.000
3	1278	1990	1287	1838	0.000
4	1712	1992	1485	1905	0.002
5	1240	1421	1200	1343	0.000
6	1298	1412	1204	1330	0.000
7	1063	1276	945	1206	0.000

Table 6.6 continued

8	963	1171	904	1084	0.000
9	2428	2680	2267	2623	0.000
10	2340	2593	2292	2539	0.000
11	2078	2369	1872	2187	0.000
12	2019	2429	2019	2380	0.003
13	1325	1474	1152	1374	0.000
14	1386	1476	1232	1407	0.000
15	1078	1259	993	1199	0.001
16	756	984	595	882	0.000
17	798	1204	709	932	0.000
18	1857	2264	1628	2120	0.000
19	2343	2646	2307	2588	0.002
20	1920	2226	1854	2077	0.000
21	1227	1361	1177	1274	0.000
22	2948	3163	2912	3079	0.000
23	4586	5210	4525	5041	0.000
24	2008	2270	1898	2166	0.000

6.8 Comparison Between VND_Proposed and Tabu Search (TS)

6.8.1 Experimental Results and Discussion

Table 6.7 shows the comparison of the soft constraint violations that occurred using the VND_Proposed and Tabu Search. As shown in Table 6.7, the VND_Proposed managed to produce a better-quality solution on average over 30 runs except for instance 8. VND_Proposed also managed to produce an improved best soft constraint violation compared to the Tabu Search for all instances except instance 11 and instance 22. The p values (< 0.05) from the t -tests for all the instances except instance 10 and instance 22 led to the rejection of the null hypothesis $H_0: \mu_{VND_Proposed} = \mu_{TabuSearch}$ and revealed a statistical difference between the means of VND_Proposed and Tabu Search. This demonstrated that

VND_Proposed outperformed Tabu Search in generating a quality solution in stage 2, and the improvement was significant. The improvement was significant as it involved the introduction of new NS: NS7 and NS8

Table 6.7: Comparison of Soft Constraint Violations between VND_Proposed and Tabu Search . $N = 30$ runs.

Instance	<i>VND_Proposed</i>		<i>TabuSearch</i>		<i>p</i> value (<i>t</i> -test)
	Best	Average	Best	Average	
1	2216	2579	2474	2682	0.000
2	2395	2645	2496	2716	0.047
3	1278	1990	1668	2120	0.032
4	1712	1992	1724	2210	0.000
5	1240	1421	1325	1482	0.004
6	1298	1412	1381	1490	0.000
7	1063	1276	1165	1322	0.046
8	963	1171	1250	1118	0.002
9	2428	2680	2558	2740	0.031
10	2340	2593	2348	2639	0.138
11	2078	2369	1887	2487	0.035
12	2019	2429	2193	2568	0.011
13	1325	1474	1381	1523	0.014
14	1386	1476	1440	1514	0.014
15	1078	1259	1105	1316	0.023
16	756	984	849	1086	0.001
17	798	1204	1043	1312	0.019
18	1857	2264	2162	2387	0.015
19	2343	2646	2486	2726	0.043
20	1920	2226	2105	2448	0.000
21	1227	1361	1250	1403	0.013
22	2948	3163	2942	3205	0.150
23	4586	5210	4831	5399	0.020
24	2008	2270	2201	2372	0.004

6.9 Conclusion

The ITC-2007 (Track 2) dataset was used to demonstrate the significant improvement obtained using the proposed algorithm. Compared to the UNIMAS dataset, HO_P failed to yield a feasible initial solution for some of the ITC-2007 (Track 2) data instances. Tabu Search (TS) was applied to the initial solution produced using HO_P to create a feasible initial solution for the ITC-2007 (Track 2) data instances. However, it has been demonstrated that HO_P outperformed HO in generating an initial solution in stage 1 in terms of the number of unallocated events, and the improvement was significant. This significance was proven using paired samples *t*-tests.

In stage 2, VND (NS5 and NS6) and VND (NS5, NS6, NS7 and NS8) were compared in terms of the average soft constraint violations. It was found that VND (NS5, NS6, NS7 and NS8) outperformed VND (NS5 and NS6), and the improvement was significant. This significance was proven using paired samples *t*-tests. It was also found that VND with a perturbation outperformed VND without a perturbation in generating a quality solution in stage 2, and the improvement was significant.

In stage 2, a comparison was done between VND (NS5, NS6, NS7 and NS8) and TS (NS5 and NS6). It was found that VND (NS5, NS6, NS7 and NS8) also managed to produce a better-quality solution on average over 30 runs except for instance 8. This shows that even though implementation of VND is simple but it still can perform better with the introduction of new NS (NS7 and NS8). VND excels at solving a wide variety of problems.

CHAPTER 7

CONCLUSION AND RECOMMENDATIONS

7.1 Introduction

This chapter summarises the research findings and highlights the contributions made by this research. This chapter also includes suggestions for future works.

7.2 Conclusion of Research

In Chapter 1, the study background, problem statement, research objectives and significance of the research were presented. The issues addressed in this study is the UCTTP at UNIMAS. The first problem is at UNIMAS, there is no practice of course pre-registration. Student course registration only takes place after the timetable has been generated. Therefore, the timetable generated cannot cater to all the students individually. The second problem is the timetable generated is based on the curriculum plan for each program. This causes the “repeaters”, those who need to repeat the course/s they failed in the previous semester, to face a certain dilemma as they cannot attend both the repeated and current semester courses as these clash. The third problem is UCTTP at UNIMAS will increase in complexity when the number of intakes at UNIMAS increases because when the number of students increases, a feasible schedule will be needed to satisfy more constraints. By doing this study, it was hoped that a suitable algorithm could be formulated to solve the timetabling problem at UNIMAS. The goal was to achieve optimisation, whereby the resources could be used effectively.

In Chapter 2, the scientific literature review of the methodologies applied to the UCTTP is presented. A classification of the methodologies is presented, based on their

chronology and the datasets used. Perspectives, trends, challenges and opportunities in the context of the UCTTP are also presented. It was observed that meta-heuristic approaches are popular among researchers, followed closely by hybrid methodologies. Hyper-heuristic approaches can also produce effective results. Another observation is that the state-of-art methodologies evident in the scientific literature are not fully utilised in real-world environments perhaps due to their limited flexibility. Real-world UCTTPs focus on the practicality of solutions at academic institutions. Even real-world UCTTPs vary in terms of requirements due to the different policies, education systems and cultures used at different institutions. Therefore, a general solution (particularly in terms of the underlying mathematical model) that fits all does not exist.

In Chapter 3, the UCTTP at the FCSIT is described. The issues identified were the curriculum-based course timetabling and the lack of student pre-registration data. The timetable is created based on each program's curriculum plan, which cannot accommodate all the students individually. In addition, all the hard and soft constraints on the problem in this study were listed. Based on these constraints, a formal presentation of the problem was presented using mathematical modelling.

The algorithm proposed in this study is presented in Chapter 4. The general framework for solving the UCTTP at the FCSIT, UNIMAS was illustrated, which consisted of two stages. The first stage is utilised to generate a feasible initial solution and the second stage is used to improve the quality of the solution. In stage 1, heuristic ordering (HO), heuristic ordering with a perturbation (HO_P) and tabu search (TS) are utilised to generate a feasible initial solution. In stage 2, variable neighbourhood descent (VND) and variable neighbourhood descent with a perturbation are utilised to improve the quality of the solution.

At the end of the chapter, tabu search algorithm is introduced as a comparison with the improved VND algorithm.

In Chapter 5, before presenting the experimental results, the features of the real-world dataset used as a testbed in this study were depicted. The data instances used were semester 1 and semester 2 for academic years 2019/2020. This chapter compares the use of different HO to find feasibility. HO (LD and (LD+LE) in descending order) managed to find a feasible solution for semester 2's instance using 31 timeslots, but only HO ((LD+LE) in descending order) managed to do so for semester 1's instance using the same number of timeslots. In comparison, the solution generated by the existing timetabling software required 48 timeslots (until 9 pm) to achieve feasibility. To further improve the HO algorithm, a perturbation was applied to the HO to find a feasible solution for semester 1's and semester 2's instances using 31 timeslots. It was found that HO with a perturbation outperformed HO in generating a feasible initial solution in stage 1, and the improvement was significant for all the instances. This significance was proven using paired samples *t*-tests. In stage 2, VND with the new NS (NS3, NS4, NS7 and NS8), in addition to the NS identified in the scientific literature (NS1, NS2, NS5 and NS6), was proposed to improve the quality of the feasible initial solution obtained in stage 1. It was found that VND (NS1, NS2, NS3, NS4, NS5, NS6, NS7, NS8) outperformed VND (NS1, NS2, NS5 and NS6), and the improvement was significant. This significance was proven using paired samples *t*-tests. It was also found that the solution generated using the proposed VND (NS1, NS2, NS3, NS4, NS5, NS6, NS7, NS8) outperformed the existing solution in terms of quality. The new NS, NS3+NS7 and NS4+NS8, were tested for their efficiency. It was found that of the new NS, NS3 + NS7 was more efficient than NS4 + NS8 in improving the quality of the solution for semester 1's instance and semester 2's instance. To help the current solution to escape from local optima,

a perturbation was applied. It was found that VND (NS1, NS2, NS3, NS4, NS5, NS6, NS7, NS8) with a perturbation outperformed VND (NS1, NS2, NS3, NS4, NS5, NS6, NS7, NS8) in generating a quality solution in stage 2, and the improvement was significant. This significance was proven using paired samples *t*-tests. The sensitivity analysis of the two-stage heuristic proposed in this research is presented in this chapter too. The final solution generated when the problem instance features had been changed (such as the number of students, number of events, number of timeslots available for allocation and number of large venues) was analysed. Semester 1's instance was chosen for the sensitivity analysis because the instance of semester 1 was larger than that of semester 2 in terms of events, students and course enrolment. The result depicts the flexibility of the simulator in solving the UCTTP at the FCSIT. If a feasible initial solution cannot be found after the addition of events or students, other techniques can be used to generate a feasible initial solution, such as adding more timeslots or using alternative algorithms. A comparison between VND (NS1, NS2, NS3, NS4, NS5, NS6, NS7, and NS8) and TS (NS1, NS2, NS5 and NS6) is made at the end of the chapter. With the introduction of the new NS (NS3, NS4, NS7, and NS8), it was discovered that VND (NS1, NS2, NS3, NS4, NS5, NS6, NS7, and NS8) outperformed TS.

In Chapter 6, the International Timetabling Competition (ITC-2007-Track 2) dataset is used to demonstrate the significant improvement obtained using the proposed algorithm. Compared to the UNIMAS dataset, HO with a perturbation failed to yield a feasible initial solution for some of the ITC-2007 (Track 2) data instances. Tabu Search (TS) was applied to the initial solution produced using HO with a perturbation to create a feasible initial solution for the ITC-2007 (Track 2) data instances. However, it was demonstrated that HO_P outperformed HO in generating an initial solution in stage 1 in terms of the number of unallocated events, and the improvement was significant. This significance was proven using

paired samples *t*-tests. In stage 2, VND (NS5 and NS6) and VND (NS5, NS6, NS7 and NS8) are compared in terms of the average soft constraint violations. It was found that VND (NS5, NS6, NS7 and NS8) outperformed VND (NS5 and NS6), and the improvement was significant. This significance was proven using paired samples *t*-tests. It was also found that VND with a perturbation outperformed VND without a perturbation in generating a quality solution in stage 2, and the improvement was significant. Stage 2 also included a comparison of TS (NS5 and NS6) and VND (NS5, NS6, NS7, and NS8). Except for instance 8, it was discovered that VND (NS5, NS6, NS7, and NS8) was likewise able to provide a better-quality solution on average over 30 runs. This indicates that even while VND implementation is simple, new NS (NS7 and NS8) can still improve performance.

7.3 Research Contributions

The contributions of this study are stated below:

- i. A comprehensive model to formally depict the UCTTP at the FCSIT, UNIMAS. The hard and soft constraints identified in this study are based on the requirements of the UCTTP at UNIMAS, as policies and regulations are unique to each institution.
- ii. The design and development of a new variable neighbourhood descent (VND) approach with new NS (NS3 NS4, NS7 and NS8) with a perturbation to solve the UCTTP at the FCSIT, UNIMAS. The approach managed to reduce the number of soft constraint violations to generate a quality solution.
- iii. Identification of the flexibility of the proposed algorithm when the problem instance features were changed. These changes were the number of events,

the number of available timeslots and large venues, as well as the number of students.

7.4 Future Work

The future of this subject could involve numerous research prospects. Firstly, the scope of the research could be broadened. Instead of merely considering lecture hours, a full schedule could be created that includes lectures, tutorials and lab sessions. The proposed algorithm could be applied to all UNIMAS faculties to further widen the scope of the research.

As mentioned in the scientific literature, population-based approaches and local search algorithms are suited to solution space exploration and exploitation, respectively. Therefore, an attempt could be made to achieve the synergy of both capabilities required to address optimisation problems. Other population-based techniques, such as genetic algorithms, could be hybridised with the VND proposed in the current study.

REFERENCES

- Abuhamdah, A., Ayob, M., Kendall, G., & Sabar, N. R. (2013). Population based Local Search for university course timetabling problems. *Applied Intelligence*, 40(1), 44–53. <https://doi.org/10.1007/s10489-013-0444-6>
- 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>
- Akkan, C., & Gülcü, A. (2018). A bi-criteria hybrid Genetic Algorithm with robustness objective for the course timetabling problem. *Computers and Operations Research*, 90, 22–32. <https://doi.org/10.1016/j.cor.2017.09.007>
- Aladag, C. H., Hocaoglu, G., & Basaran, M. A. (2009a). The effect of neighborhood structures on tabu search algorithm in solving course timetabling problem. *Expert Systems with Applications*, 36(10), 12349–12356. <https://doi.org/10.1016/j.eswa.2009.04.051>
- Albalooshi, F., & Shatnawi, S. (2010). Online academic advising support. *Technological Developments in Networking, Education and Automation*, 25–29. <https://doi.org/10.1007/978-90-481-9151-2-5>
- Almeida, M. W. S., Medeiros, J. P. S., & Oliveira, P. R. (2016). Solving the academic timetable problem thinking on student needs. *Proceedings - 2015 IEEE 14th International Conference on Machine Learning and Applications, ICMLA 2015*, 673–676. <https://doi.org/10.1109/ICMLA.2015.184>
- Assi, M., Halawi, B., & Haraty, R. A. (2018). Genetic Algorithm Analysis using the Graph Coloring Method for Solving the University Timetable Problem. *Procedia Computer*

- Science*, 126, 899–906. <https://doi.org/10.1016/j.procS.2018.08.024>
- Avella, P., Boccia, M., Viglione, S., & Vasilyev, I. (2019). A Local Branching {MIP} Heuristic for a Real-World Curriculum-Based Course Timetabling Problem. In *Mathematical Optimization Theory and Operations Research* (pp. 438–451). Springer International Publishing. https://doi.org/10.1007/978-3-030-33394-2_34
- Aycan, E., & Ayav, T. (2009). Solving the course scheduling problem using simulated annealing. *2009 IEEE International Advance Computing Conference, March*, 462–466.
- Azad, S. K., & Hasançebi, O. (2014). An elitist self-adaptive step-size search for structural design optimization. *Applied Soft Computing*, 19, 226–235.
- Babaei, H., Karimpour, J., & Hadidi, A. (2015). A survey of approaches for university course timetabling problem. *Computers and Industrial Engineering*, 86, 43–59. <https://doi.org/10.1016/j.cie.2014.11.010>
- Badoni, R. P., & Gupta, D. K. (2016). A new algorithm based on students groupings for university course timetabling problem. *2015 2nd International Conference on Recent Advances in Engineering and Computational Sciences, RAECS 2015, December*. <https://doi.org/10.1109/RAECS.2015.7453340>
- Bagger, N. C. F., Desaulniers, G., & Desrosiers, J. (2019). Daily course pattern formulation and valid inequalities for the curriculum-based course timetabling problem. *Journal of Scheduling*, 22(2), 155–172. <https://doi.org/10.1007/s10951-018-0582-0>
- Bardadym, V. A. (1996). Computer-aided school and university timetabling: The new wave. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 1153, 22–45. https://doi.org/10.1007/3-540-61794-9_50
- Borchani, R., Elloumi, A., & Masmoudi, M. (2017). Variable neighborhood descent search

- based algorithms for course timetabling problem: Application to a Tunisian University. *Electronic Notes in Discrete Mathematics*, 58, 119–126.
<https://doi.org/10.1016/j.endm.2017.03.016>
- Budiono, T. A., & Wong, K. W. (2012, June). A pure graph coloring constructive heuristic in timetabling. *2012 International Conference on Computer & Information Science (ICCIS)*. <https://doi.org/10.1109/iccisci.2012.6297259>
- Burke, E. (1997). Automated University Timetabling: The State of the Art. *The Computer Journal*, 40(9), 565–571. <https://doi.org/10.1093/comjnl/40.9.565>
- Burke, Edmund, Curtois, T., Post, G., Qu, R., Veltman, B., Burke, C. E., Curtois, T., Post, G., Qu, R., & Veltman, B. (2005). *University of Nottingham Jubilee Campus Computer Science Technical Report No . NOTTCS-TR-2005-9 A Hybrid Heuristic Ordering and Variable Neighbourhood Search for the Nurse Rostering Problem A Hybrid Heuristic Ordering and Variable Neighbourhood Search for.*
- Burke, Edmund K., & Bykov, Y. (2017). The late acceptance Hill-Climbing heuristic. *European Journal of Operational Research*, 258(1), 70–78.
<https://doi.org/10.1016/j.ejor.2016.07.012>
- Burke, Edmund 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>
- Burke, Edmund K., McCollum, B., Meisels, A., Petrovic, S., & Qu, R. (2007). A graph-based hyper-heuristic for educational timetabling problems. *European Journal of Operational Research*, 176(1), 177–192. <https://doi.org/10.1016/j.ejor.2005.08.012>
- Burke, Edmund Kieran, & Petrovic, S. (2002). Recent research directions in automated timetabling. *European Journal of Operational Research*, 140(2), 266–280.

[https://doi.org/10.1016/S0377-2217\(02\)00069-3](https://doi.org/10.1016/S0377-2217(02)00069-3)

- Ceschia, S., Di Gaspero, L., & Schaerf, A. (2012). Design, engineering, and experimental analysis of a simulated annealing approach to the post-enrolment course timetabling problem. *Computers and Operations Research*, 39(7), 1615–1624. <https://doi.org/10.1016/j.cor.2011.09.014>
- Chen, R. M., & Shih, H. F. (2013). Solving university course timetabling problems using constriction particle swarm optimization with local search. *Algorithms*, 6(2), 227–244. <https://doi.org/10.3390/a6020227>
- Chiarandini, M., Birattari, M., Socha, K., & Rossi-Doria, O. (2006). An effective hybrid algorithm for university course timetabling. *Journal of Scheduling*, 9(5), 403–432. <https://doi.org/10.1007/s10951-006-8495-8>
- de Werra, D. (1985). An introduction to timetabling. *European Journal of Operational Research*, 19(2), 151–162. [https://doi.org/10.1016/0377-2217\(85\)90167-5](https://doi.org/10.1016/0377-2217(85)90167-5)
- Deb, K., & Jain, H. (2014). An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box Constraints. *{IEEE} Transactions on Evolutionary Computation*, 18(4), 577–601. <https://doi.org/10.1109/tevc.2013.2281535>
- Dorigo, M., & Stutzle, T. (2009). Ant colony optimization: Overview and recent advances. *International Series In Operations Research and Management Science*, 272, 311–351.
- Drake, J. H., Kheiri, A., Özcan, E., & Burke, E. K. (2020). Recent advances in selection hyper-heuristics. *European Journal of Operational Research*, 285(2), 405–428. <https://doi.org/10.1016/j.ejor.2019.07.073>
- Erdeniz, S. P., & Felfernig, A. (2018). *OCSH : Optimized Cluster Specific Heuristics for The University Course Timetabling Problem*. 0–5.

- Ghalia, M. Ben. (2008). Particle swarm optimization with an improved exploration-exploitation balance. *Midwest Symposium on Circuits and Systems*, 759–762. <https://doi.org/10.1109/MWSCAS.2008.4616910>
- Goh, S. L., Graham, G., Sabar, N. R., & Abdullah, S. (2020). An effective hybrid local search approach for the post enrolment course timetabling problem. *OPSEARCH*. <https://doi.org/10.1007/s12597-020-00444-x>
- Goh, S. L., Kendall, G., & Sabar, N. R. (2017). Improved local search approaches to solve the post enrolment course timetabling problem. *European Journal of Operational Research*, 261(1), 17–29. <https://doi.org/10.1016/j.ejor.2017.01.040>
- Goh, S. L., Kendall, G., & Sabar, N. R. (2019). Simulated annealing with improved reheating and learning for the post enrolment course timetabling problem. *Journal of the Operational Research Society*, 70(6), 873–888. <https://doi.org/10.1080/01605682.2018.1468862>
- Gora, W., Lach, G., Lübke, J., Pfeiffer, O., Zorn, E., & Jeschke, S. (2010). *Management and Optimal Distribution of Large Student Numbers*. 1891–1896.
- Gülcü, A., & Akkan, C. (2020). Robust university course timetabling problem subject to single and multiple disruptions. *European Journal of Operational Research*, 283(2), 630–646. <https://doi.org/10.1016/j.ejor.2019.11.024>
- Gunawan, A., Ng, K. M., & Poh, K. L. (2012). A hybridized Lagrangian relaxation and simulated annealing method for the course timetabling problem. *Computers and Operations Research*, 39(12), 3074–3088. <https://doi.org/10.1016/j.cor.2012.03.011>
- Habashi, S. S., Yousef, A. H., Salama, C., & Fahmy, H. M. A. (2018). Adaptive Diversifying Hyper-Heuristic Based Approach for Timetabling Problems. *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference*

(IEMCON), 259–266.

- Hansen, P., Mladenović, N., Brimberg, J., & Pérez, J. A. M. (2018). Variable Neighborhood Search. In *International Series in Operations Research & Management Science* (Vol. 272). Springer New York LLC. https://doi.org/10.1007/978-3-319-91086-4_3
- Harada, T., & Alba, E. (2020). Parallel genetic algorithms: a useful survey. *ACM Computing Surveys (CSUR)*, 53(4), 1–39.
- Ilyas, R., & Iqbal, Z. (2015, June). Study of hybrid approaches used for university course timetable problem ({UCTP}). *2015 {IEEE} 10th Conference on Industrial Electronics and Applications ({ICIEA})*. <https://doi.org/10.1109/iciea.2015.7334198>
- Kampke, E. H., Scheideger, L. M., Mauri, G. R., & Boeres, M. C. S. (2019). A Network Flow Based Construction for a GRASP + SA Algorithm to Solve the University Timetabling Problem. In *International Conference on Computational Science and Its Applications*, 215–231. <https://doi.org/10.1007/978-3-030-24302-9>
- Kiefer, A., Hartl, R. F., & Schnell, A. (2017). Adaptive large neighborhood search for the curriculum-based course timetabling problem. *Annals of Operations Research*, 252(2), 255–282. <https://doi.org/10.1007/s10479-016-2151-2>
- Lewis, Rhydian. (2010). A time-dependent metaheuristic algorithm for post enrolment-based course timetabling. *Annals of Operations Research*, 194(1), 273–289. <https://doi.org/10.1007/s10479-010-0696-z>
- Lewis, Rhydian. (2008). A survey of metaheuristic-based techniques for University Timetabling problems. *OR Spectrum*, 30(1), 167–190. <https://doi.org/10.1007/s00291-007-0097-0>
- Lewis, Rhydian, & Paechter, B. (2007). Finding feasible timetables using group-based operators. *IEEE Transactions on Evolutionary Computation*, 11(3), 397–413.

- <https://doi.org/10.1109/TEVC.2006.885162>
- Lindahl, M., Mason, A. J., Stidsen, T., & Sørensen, M. (2018). A strategic view of University timetabling. *European Journal of Operational Research*, 266(1), 35–45.
<https://doi.org/10.1016/j.ejor.2017.09.022>
- Liu, Y., Zhang, D., & Chin, F. Y. L. (2011). A clique-based algorithm for constructing feasible timetables. *Optimization Methods and Software*, 26(2), 281–294.
<https://doi.org/10.1080/10556781003664739>
- Lü, Z., & Hao, J. K. (2010). Adaptive Tabu Search for course timetabling. *European Journal of Operational Research*, 200(1), 235–244. <https://doi.org/10.1016/j.ejor.2008.12.007>
- Matias, J. B., Fajardo, A. C., & Medina, R. P. (2019). A hybrid genetic algorithm for course scheduling and teaching workload management. *2018 IEEE 10th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management, HNICEM 2018*, 1–6.
<https://doi.org/10.1109/HNICEM.2018.8666332>
- McCollum, B. (2006). A perspective on bridging the gap between theory and practice in university timetabling. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 3867 LNCS, 3–23. https://doi.org/10.1007/978-3-540-77345-0_1
- Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey Wolf Optimizer. *Advances in Engineering Software*, 69, 46–61.
<https://doi.org/https://doi.org/10.1016/j.advengsoft.2013.12.007>
- Muklason, A., Irianti, R. G., & Marom, A. (2019). Automated course timetabling optimization using tabu-variable neighborhood search based hyper-heuristic algorithm. *Procedia Computer Science*, 161, 656–664.

<https://doi.org/10.1016/j.procs.2019.11.169>

- Nagata, Y. (2018). Random partial neighborhood search for the post-enrollment course timetabling problem. *Computers and Operations Research*, 90, 84–96. <https://doi.org/10.1016/j.cor.2017.09.014>
- Nothegger, C., Mayer, A., Chwatal, A., & Raidl, G. R. (2012). Solving the post enrolment course timetabling problem by ant colony optimization. *Annals of Operations Research*, 194(1), 325–339. <https://doi.org/10.1007/s10479-012-1078-5>
- Oude Vrielink, R. A., Jansen, E. A., Hans, E. W., & van Hillegersberg, J. (2019). Practices in timetabling in higher education institutions: a systematic review. *Annals of Operations Research*, 275(1), 145–160. <https://doi.org/10.1007/s10479-017-2688-8>
- Pillay, N., & Özcan, E. (2019). Automated generation of constructive ordering heuristics for educational timetabling. *Annals of Operations Research*, 275(1), 181–208. <https://doi.org/10.1007/s10479-017-2625-x>
- Prabodanie, R. A. R. (2017). An Integer Programming Model for a Complex University Timetabling Problem: A Case Study. *Industrial Engineering and Management Systems*, 16(1), 141–153. <https://doi.org/10.7232/iems.2017.16.1.141>
- Ross, P., & Marin-Blazquez, J. G. (2005). Constructive hyper-heuristics in class timetabling. In *Proc. IEEE Congr. Evol. Comput.* (pp. 1493–1500)
- Sörensen, K., & Glover, F. (1986). *METAHEURISTICS I* Kenneth Sörensen University of Antwerp, Belgium Fred Glover University of Colorado and OptTek Systems, Inc., USA.
- Sabar, N. R., Ayob, M., Kendall, G., & Qu, R. (2012). A honey-bee mating optimization algorithm for educational timetabling problems. *European Journal of Operational Research*, 216(3), 533–543. <https://doi.org/10.1016/j.ejor.2011.08.006>
- Sabar, N. R., Turkey, A., Song, A., & Sattar, A. (2020). An evolutionary hyper-heuristic to

- optimise deep belief networks for image reconstruction. *Applied Soft Computing*, 97, 105510.
- Schaerf, A. (1999). Survey of automated timetabling. *Artificial Intelligence Review*, 13(2), 87–127. <https://doi.org/10.1023/A:1006576209967>
- Schindl, D. (2019). Optimal student sectioning on mandatory courses with various sections numbers. *Annals of Operations Research*, 275(1), 209–221. <https://doi.org/10.1007/s10479-017-2621-1>
- Shaker, K., & Abdullah, S. (2010). Controlling Multi Algorithms Using Round Robin for University Course Timetabling Problem. In *Database Theory and Application, Bio-Science and Bio-Technology* (pp. 47–55). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-17622-7_6
- Shatnawi, S. M., Albalooshi, F., & Rababa, K. (2012). Generating Timetable and Students schedule based on data mining techniques. *International Journal of Engineering Research and Applications*, 2(4), 1638–1644.
- Song, T., Liu, S., Tang, X., Peng, X., & Chen, M. (2018). An iterated local search algorithm for the University Course Timetabling Problem. *Applied Soft Computing Journal*, 68, 597–608. <https://doi.org/10.1016/j.asoc.2018.04.034>
- Sze, S. N., Bong, C. L., Chiew, K. L., Tiong, W. K., & Bolhassan, N. A. (2017). Case study: University lecture timetabling without pre-registration data. *Proceedings of the 2017 IEEE International Conference on Applied System Innovation: Applied System Innovation for Modern Technology, ICASI 2017, c*, 732–735. <https://doi.org/10.1109/ICASI.2017.7988533>
- 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(May 2020), 113943. <https://doi.org/10.1016/j.eswa.2020.113943>
- Tan, J. S., Say, T., Goh, 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>
- Teoh, C. K., Wibowo, A., & Ngadiman, M. S. (2015). Review of state of the art for metaheuristic techniques in Academic Scheduling Problems. *Artificial Intelligence Review*, 44(1), 1–21. <https://doi.org/10.1007/s10462-013-9399-6>
- Thepphakorn, T., & Pongcharoen, P. (2020). Performance Improvement Strategies on Cuckoo Search Algorithms for Solving the University Course Timetabling Problem. *Expert Systems With Applications*, 113732. <https://doi.org/10.1016/j.eswa.2020.113732>
- Turabieh, H., Abdullah, S., McCollum, B., & McMullan, P. (2010). Fish Swarm Intelligent Algorithm for the Course Timetabling Problem. In *Lecture Notes in Computer Science* (pp. 588–595). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-16248-0_80
- Turky, A., Sabar, N. R., Dunstall, S., & Song, A. (2020). Hyper-heuristic local search for combinatorial optimisation problems. *Knowledge-Based Systems*, 205, 106264.
- Vianna, D. S., Martins, C. B., Lima, T. J., Vianna, M. de F. D., & Meza, E. B. M. (2020). Hybrid VNS-TS heuristics for University Course Timetabling Problem. *Brazilian Journal of Operations & Production Management*, 17(1), 1–20. <https://doi.org/10.14488/bjopm.2020.014>
- Wang, B., Geng, Y., & Zhang, Z. (2019). Applying genetic algorithm to university classroom arrangement problem. *Journal of Physics: Conference Series*, 1325(1). <https://doi.org/10.1088/1742-6596/1325/1/012157>

APPENDICES

Appendix A: Journal Publications

1. **Chen, M. C.**, Sze, S. N., Goh, S. L., & Lau, S. P. (2021). A Hybrid of Heuristic Orderings and Variable Neighbourhood Descent for a Real Life University Course Timetabling Problem. *International Journal of Systematic Innovation*, 6(5), 1–10.
[https://doi.org/10.6977/IJoSI.202109_6\(5\).0001](https://doi.org/10.6977/IJoSI.202109_6(5).0001)
2. **Chen, M. C.**, Sze, S. N., Goh, S. L., Sabar, N. R., & Kendall, G. (2021). A Survey of University Course Timetabling Problem: Perspectives, Trends and Opportunities. *IEEE Access*, 9, 106515–106529. <https://doi.org/10.1109/ACCESS.2021.3100613>
3. **Chen, M. C.**, Sze, S. N., Goh, S. L., & Lau, S. P. (2022). Investigation of Heuristic Orderings with a Perturbation for Finding Feasibility in Solving Real- World University Course Timetabling Problem. *Proceedings of the 2022 International Conference on Digital Transformation and Intelligence (ICDI)*, 168–173.
[https:// doi: 10.1109/ICDI57181.2022.10007172](https://doi.org/10.1109/ICDI57181.2022.10007172)