

PARTICLE SWARM OPTIMIZATION TO  
SOLVE THE ITC2007 EXAMINATION  
TIMETABLING PROBLEM

EE JUN JIANG

UNIVERSITY MALAYSIA PAHANG

PARTICLE SWARM  
EXAMINA



THE ITC 2007

EE JUN JIANG

THESIS SUBMITTED IN FULLFILLMENT OF THE DEGREE OF COMPUTER  
SCIENCE (SOFTWARE ENGINEERING)

FACULTY OF COMPUTER SYSTEM AND SOFTWARE ENGINEERING  
UNIVERSITY MALAYSIA PAHANG

2014

849801



## UNIVERSITI MALAYSIA PAHANG

## BORANG PENGESAHAN STATUS TESIS

JUDUL: Particle Swarm Optimization to solve the ITC2007 examination timetabling problem

SESI PENGAJIAN: Semester 1 2014/2015

SAYA EE JUN JIANG (HURUF BESAR)

Mengaku membenarkan tesis/laporan PSM ini disimpan di Perpustakaan Universiti Malaysia Pahang dengan syarat-syarat kegunaan seperti berikut:

1. Tesis/Laporan adalah hakmilik Universiti Malaysia Pahang.
2. Perpustakaan Universiti Malaysia Pahang dibenarkan membuat salinan untuk tujuan pengajian sahaja.
3. Perpustakaan dibenarkan membuat salinan tesis ini sebagai bahan pertukaran antara institut pengajian tinggi.
4. \*\*Sila tandakan (v)



SULIT

(Mengandungi maklumat yang berdarjah keselamatan atau kepentingan Malaysia seperti yang termaktub di dalam AKTA RAHSIA RASMI 1972) \*



TERHAD

(Mengandungi maklumat TERHAD yang telah ditentukan oleh organisasi/badan di mana penyelidikan dijalankan) \*



TIDAK TERHAD

Disahkan Oleh

Penyelia: MOHD NIZAM FAHAR

Alamat tetap:  
146, Jalan Salad 4  
Taman Salad 09600  
Lunas, Kedah  
 Tarikh: 23/12/2014

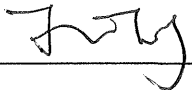
Tarikh: 23/12/2014

\*Sila lampirkan surat daripada pihak berkuasa/organisasi berkenaan dengan menyatakan sekali sebab dan tempoh tesis/laporan ini perlu dikelaskan sebagai SULIT atau TERHAD.

**DECLARATION**

I hereby declare that the work in this thesis is my own except for quotations and summaries which have been duly acknowledged.

15 MAY 2014



---

Ee Jun Jiang  
CB 11088

### SUPERVISOR'S DECLARATION

"I hereby declare that I have read this thesis and my opinion this thesis is sufficient in terms of scope and quality for the submission of PSM 2, Degree in Computer Science (Software Engineering)"

Signature

:  .....

Supervisor

: Encik Mohd Nizam Bin Mohmad Kahar

Date

: 23/12/2014 .....

## **ACKNOWLEDGEMENTS**

First, thank god who is always with me for giving me strength and bless me to finish this work. Next, I would like to express my sincere thanks to my supervisor Dr. Mohd Nizam bin Mohmad Kahar for all his guidance and encouragement. I am greatly appreciated his concern and support at all time.

Lastly, thanks my family for their support, caring and love as a motivation to me and my academics. Not forgetting my friends that gave me helps when I face problem.

## ABSTRACT

The educational institute has been annoyed by timetabling problem for many years. The timetabling is difficult due to large amount of subject/exams to be allocated into certain timeslots at the same time fulfill all the hard constraints and some of the soft constraints (constraints are requirements of the timetable). In the timetabling research, there are various techniques/ algorithm has been used to solve the problem. However, most of them are applied to solve simpler timetabling problem which doesn't consider room capacity as constraints (or simplified the room capacity factor).

In this thesis, an examination timetabling dataset that consider room capacity will be used which named Seconds International Timetabling Competition (ITC 2007). ITC 2007 has very well defined constraints and it is more completed compared to other dataset.

One chosen technique which is Particle Swarm Optimization will be used to solve the ITC 2007 examination timetabling problem.

## ABSTRAK

Institut pendidik telah bertahun-tahun diganggu masalah membuat jadual waktu. Membuat jadual waktu ini amat sukar kerana jumlah mata pelajaran / peperiksaan yang akan diperuntukan ke dalam slot masa tertentu disampingan mencapai kekangan keras and kekangan lembut. Dalam kajian jadual waktu, terdapat pelbagai teknik/ algoritma telah digunakan untuk menyelesaikan masalah tersebut. Walaubagimanapun, sebahagian besar kajian adalah digunakan untuk menyelesaikan masalah jadual waktu yang lebih mudah, iaitu kajian yang tidak menganggap kapasiti bilik sebagai kekangan (atau mempermudah faktor kapasiti bilik).

Dalam tesis ini, satu dataset jadual waktu peperiksaan yang bernama International Timetabling Competition (ITC 2007) akan digunakan. ITC 2007 mempunyai kekangan yang baik ditakrif dan ia lebih lengkap berbanding dengan dataset lain. Salah satu teknik yang dipilih adalah Particle Swarm Optimization akan digunakan untuk menyelesaikan masalah jadual waktu peperiksaan ITC 2007.



## TABLE OF CONTENT

	<b>PAGE</b>
<b>DECLARATION</b>	iv
<b>SUPERVISOR's DECLARATION</b>	v
<b>ACKNOWLEDGEMENTS</b>	vi
<b>ABSTRACT</b>	vii
<b>ABSTRAK</b>	viii
<b>CONTENTS</b>	ix
<b>LIST OF TABLES</b>	xi
<b>LIST OF FIGURES</b>	xii

<b>SECTION</b>	<b>CONTENT</b>	<b>PAGE</b>
<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
1.1	Background of Study	1
1.2	Problem Statement	2
1.3	Research Objective	3
1.4	Research Scope	3
1.5	Thesis Organization	3
<b>2</b>	<b>LITERATURE REVIEW</b>	<b>4</b>
2.1	Overview of Timetabling	4
2.2	University timetabling problems	5
2.3	Examination timetabling	6
	2.3.1 Main objective of examination timetabling and its constraints	7
	2.3.2 Capacitated and un-capacitated problem in examination timetabling	8
2.4	Examination timetabling dataset	8
	2.4.1 University of Toronto dataset	9
	2.4.2 Dataset of University of Melbourne	10
	2.4.3 Dataset of University of Nottingham	11
	2.4.4 Dataset of University Kebangsaan Malaysia(UKM)	11
	2.4.5 Dataset of University Teknologi MARA(UiTM)	12
	2.4.6 Second International Timetabling Competition (ITC2007) dataset	13
	2.4.7 Summary of dataset	16
2.5	Methodologies applied to the examination timetabling problem	18
	2.5.1 Hill Climbing (HC)	18
	2.5.2 Tabu search (TS)	19
	2.5.3 Simulated Annealing (SA)	20
	2.5.4 Great Deluge Algorithm (GDA)	21
	2.5.5 Genetic Algorithm (GA)	23
	2.5.6 Ant Colony Optimization(ACO)	24
	2.5.7 Graph Heuristic (GH)	25
	2.5.8 Particle Swarm Optimization (PSO)	27

2.6	Conclusion	29
<b>3</b>	<b>METHODOLOGY</b>	<b>30</b>
3.1	Introduction	30
3.2	ITC 2007 : Examination Track	30
3.3	Problem Formulation	31
3.4	Particle Swarm Optimization to solve ITC 2007 examination timetabling problem	35
3.5	Conclusion	37
<b>4</b>	<b>DESIGN AND IMPLEMENTATION</b>	<b>38</b>
4.1	Project Implementation	38
	4.1.1 Display the information of the ITC2007 dataset	38
	4.1.2 Coding	42
	4.1.2.1 Calculate the information	43
	4.1.2.2 Calculate conflict matrix, matrix density, number of exam conflicts and student conflicts	44
	4.1.2.3 Sorting Exams with LE, LWD and LD	44
	4.1.2.4 Sorting Method	46
4.2	Solving ITC 2007	48
	4.2.1 Initialization of timetable with graph heuristic	48
	4.2.2 Improving the quality of initial solution by PSO	50
4.3	Conclusion	53
<b>5</b>	<b>RESULT AND DISCUSSION</b>	<b>54</b>
5.1	Result of penalty value from different exam sorting method	54
5.2	Performance of PSO with different population size	56
5.3	System overview	57
5.4	Conclusion	58
<b>6</b>	<b>CONCLUSION</b>	<b>59</b>
6.1	Introduction	59
6.2	Result analysis	59
6.3	Future work	60
	<b>REFERENCES</b>	<b>61</b>
	<b>APPENDIX</b>	<b>64</b>

## LIST OF TABLES

NO	TABLE TITLE	PAGE
2.1	Examples of constraints of the course timetabling problems	6
2.2	Example of constraints for the examination timetabling problems	7
2.3	Toronto Dataset	10
2.4	University of Melbourne datasets	11
2.5	University of Nottingham datasets	11
2.6	University Kebangsaan Malaysia datasets (UKM06-01)	12
2.7	Capacity of dataset UKM06-01	12
2.8	University Teknologi Malaysia (UiTM) dataset	13
2.9	International Timetabling competition dataset	14
2.10	Hard constraints of ITC2007	14
2.11	Soft constraints of ITC2007	15
2.12	Summary of datasets	16
5.1	Result of penalty value from different sorting method	54
5.2	Statistical view of LE and LD	55
5.3	Result of performance test for PSO	56
5.4	Statistical view of performance test for PSO	56

## LIST OF FIGURES

NO	FIGURE TITLE	PAGE
2.1	The Hill Climbing procedures	19
2.2	GDA procedures	22
2.3	Genetic Algorithm procedure	24
2.4	Common ordering strategies of graph heuristic	26
2.5	Procedure of PSO	28
4.1	Exam.txt	39
4.2	Room.txt	39
4.3	Period.txt	40
4.4	periodHC.txt	40
4.5	roomHC.txt	41
4.6	conflictMatrix.txt	41
4.7	institutionalWeight.txt	42
4.8	Coding for txt file loading	43
4.9	The information of dataset after calculated	43
4.10	Coding of displaying the dataset information	44
4.11	Exam sorted with ascending LE	45
4.12	Exam sorted with ascending LD	45
4.13	Exam sorted with ascending LWD	46
4.14	QuickSort class	47
4.15	Calling quicksort function to sort LD	47
4.16	Flowchart of generating an ITC 2007 timetable by using graph heuristic	49
4.17	Flowchart of timetable quality improvement by using PSO	51

## CHAPTER 1

### INTRODUCTION

#### 1.1 Background of Study

Examination timetabling problem has annoying educational institute for many years and it has been widely studied by many researchers. It concerns with the distribution of the university exams and also the timeslot.

Examination timetabling problems can be divided into 2 types: capacitated and un-capacitated. The researcher can solve the un-capacitated examinations problem with their techniques or algorithm easily and quick. However, this is difficult for capacitated problem because it consider the room capacity. Room capacity will be the hard constraints of capacitated examination problem and it makes capacitated problem look like real world.

There are many solutions and algorithms can be used to solve this problem. While solving the timetabling problem, researcher have to fulfill certain constrains. The constraints can be divided into 2 types which are hard constraints and soft constraints. Hard constraints are the requirement of the timetable and it must be achieve, otherwise the timetable is not usable and will consider as a failure. An example of hard constraint is *amount of student should not exceed room capacity*. While soft constraints are the rules that are not necessary to fulfill. However, if you achieve as much soft constraint as possible, it will enhance the quality of the timetable. An example of the soft constraint is *exams should be spread as evenly as possible*.

There is lesser researcher work on capacitated timetabling problem than un-capacitated timetabling problem due to the difficulty. Another reason would be lack of dataset of capacitated problem. The researchers who are working on capacitated timetabling problem are concerned of room number and room's size because these constraints represent the increasing of complexity of the timetable.

We will work on capacitated problem which is the *International Timetabling Competition 2007 (ITC2007)*. There are some constraints in ITC 2007 which will be explained in detail later. *Particle Swarm Optimization* is chosen as the technique to solve ITC 2007.

## 1.2 Problem statement

Many universities will agree on how hard it is to conduct an examination timetable. However, it can be solved by using the algorithm in computer science.

The examination timetabling problem is one of the common timetabling problem. However, a lot of research of examination timetabling has been made with the un-capacitated dataset such as University of Toronto dataset. This doesn't really resemble the real world timetabling situation. There are also some capacitated dataset such as University of Nottingham and University of Melbourne dataset they combined the every room capacity as one. In real world, we should separate the room capacity individually. Thus, the research is not really practical.

Our work will be focus on solving the *ITC2007* examination timetabling dataset practically. This capacitated dataset has several constraints that resemble real world situation.

### **1.3 Research objective**

This research will be achieving three objectives:

1. To study on the examination track of the Second International Timetabling Competition (ITC2007).
2. To implement Particle Swarm Optimization method in solving the examination timetabling problem that satisfies all the constraints.
3. To validate and verify the solution produced using Particle Swarm Optimization whether it satisfies all the constraints.

### **1.4 Research scope**

In our research, we are going to study ITC2007 timetabling dataset. This dataset has some constraints including room capacity and the room number. We are implementing Particle Swarm Optimization to develop a timetable for the dataset.

### **1.5 Thesis Organization**

This research consists of 7 chapters. Chapter 1 is Introduction. Chapter 2 is literature review. Chapter 3 is methodology. Chapter 4 and 5 are design and implementation of the system. Chapter 6 is the result of the research and discussion of the research. Last, Chapter 7 is conclusion of the research.

## **CHAPTER 2**

### **LITERATURE REVIEW**

#### **2.1 Overview of timetabling**

Timetabling is concerned about how the subject are allocated within the limited timeslots. There are many kinds of timetable scheduling problems in the daily life, such as examination, lecture, and transportation timetable. For all of these timetabling problems, there are constraints that needed to be satisfied while solving these timetabling problems.

The constraints of timetabling can be categorized into hard constraints and soft constraints. Hard constraints are rules the timetable must follow. A timetable is usable only if all hard constraints are achieved. Soft constraints are not necessary to be followed but it is encouraged to fulfill as much as possible. This will enhance the quality of the timetable. However, it cost us more effort to handle soft constraints.



## 2.2 University timetabling problems

The very common timetabling problems in University are course timetabling and exam timetabling. They both take a lot of time and manpower to conduct manually. Prevent allocating students to sit two examinations/courses at the same time is the first priority of timetabling. However, examination timetabling and course timetabling have some difference between each other. First, course timetabling and exam timetabling have significant difference in their constraints. The differences of the constraints will be presented on Table 2.1 and Table 2.2.

Other than the constraints, we can differentiate examination timetabling and course timetabling by the way of construction. There are 3 noticeable construction can help us which are environment of the process, modeling and scheduling instances. For exam timetable, it is conducted by the academic office while course timetable is conducted independently by school. Their process environment is different. In modeling, exam timetable was conducted based on amount of student that are taking the exam. While course timetable was conducted based on amount of student and amount of course. For the scheduling instances, exam timetable was conducted based on offered course and course timetable was conducted based on offered course as well as lecturer's available session.

Many Universities allows student to arrange the timetable themselves by system Open Registration, this increased the complexity of scheduling the timetable. The complexity is relevant to the degree of freedom given to student to arrange the timetable themselves.

**Table 2.1** Some constraints of Course Timetabling problem**Hard constraints**

1. No student or lecturer is allowed to participate in more than one course in the same time.
2. A classroom can only have one course at a timeslots.
3. The participant number should less than the space available in the classroom.
4. The classroom assigned to the course should satisfy the features required by the course.

**Soft constraints**

5. Students should have more than one course in each day.
6. A course should schedule to the final timeslot of each day.

**2.3 Examination timetabling**

In this research, my main focus is on solving Examination timetabling problem. Examination timetable plays a huge role in educational institution. It should be conducted very carefully in order to avoid clashing schedule and cause trouble to student or lecturer.

The main task of examination tabling is to allocate the examinations into certain rooms and timeslots. At the same time, the timetable should follow all the constraints. Table 2.2 shows some examples of constraints in examination timetabling problems.

**Table 2.2** Some constraints of Examination Timetabling problems

<p><b>Hard constraints</b></p> <ol style="list-style-type: none"> <li>1. No exams with common resources (student) assigned simultaneously.</li> <li>2. The resources of the examinations should be enough (size of the room, number of room).</li> </ol> <p><b>Soft constraints</b></p> <ol style="list-style-type: none"> <li>3. Spreading exams as even as possible.</li> <li>4. All the examination should be scheduled and schedule the largest exams as soon as possible.</li> <li>5. Limit the number of student in any timeslot.</li> <li>6. Examination that has similar time length can be assigned in a similar room.</li> <li>7. Time requirements (exams (not) to be in certain timeslots).</li> <li>8. Examinations to be consecutive.</li> <li>9. Ordering (precedence) of exams need to be satisfied.</li> <li>10. Examination that having conflict in same day should be located nearby.</li> </ol>
---

### 2.3.1 Main objective of examination timetabling and its constraints.

There are a lot of examination timetabling constraints because different institution might have different request. More than that, the person who related to timetable always have different preference on the timetable. For example, student will demand examination being spread evenly in order to give them times to do their revision.

These constraints should be achieved if possible in order to create high quality timetable.

### **2.3.2 Capacitated & un-capacitated problem in examination timetabling.**

In the timetabling field, most of the researchers are working on the un-capacitated problem. This is because un-capacitated problem doesn't consider room capacity as constraints so it is easier to study and investigate. However, capacitated problem more resemble the real world situation because it is impossible for an educational institute to have unlimited space for exam. Capacitated timetabling problem is so much difficult than un-capacitated timetabling problem.

Capacitated problem requires more comprehensive data such as examination list, student list. This extra information is difficult to gather. And the main cause to the difficulty of scheduling timetable is lack of available room.

Burke, Newall and Weare, (1996) modified Toronto dataset to make it more resemble the real world timetabling situation. The modified Toronto dataset includes an overall capacity as if all exams were placed in a one big room. This represents the simplification of the timetabling problem since we would have to consider only one room's capacity.

### **2.4 Examination timetabling dataset.**

There are a lot of timetabling datasets in the community. The most common dataset are University of Toronto, University of Nottingham and University of Melbourne. Other than that, we have UKM examination datasets and UiTM examination dataset. In 2008, McCollum et al (2008) introduced the Second International Timetabling Competition (ITC 2007). This dataset have more realistic problem as it consider the room capacity compared to other datasets.

There are a lot of research on the un-capacitated problem, mainly concentrate on the algorithms and the performance of that algorithm. The result will be observed to see whether the algorithm is capable to produce the timetable effectively and quick enough.

Most of the researchers on un-capacitated timetabling are not dealing with all aspects of the timetabling problem, they work on simplified examination timetabling problem instead. The simplified examination timetabling problem only includes some very common hard constraints and soft constraints.

#### **2.4.1 University of Toronto dataset**

The Toronto dataset have 13 real-world exam timetabling problems. To allow genuine comparison between the scientific communities, problem instances of the Toronto dataset are classified into I and also II by Qu et al, (2009). The University of Toronto dataset was introduced by Carte, Laporte and Lee, (1996). They investigated two variants of the objectives with the purpose to minimize the number of timeslots needed for the problem and to minimize the sum of approximate costs per student. In 2001, Di Gaspero and Schaerf used Tabu Search with only consider constraints conflict to solve this Toronto dataset. Burke, Newall and Weare, (1996) had added some new aim to the Toronto dataset. They made count of maximum amount of the room capacity per timeslot and second-order conflict of same day constraints. Merlot et al. (2003) solved Toronto dataset by using the hybridization of Constraint Programming, Hill Climbing and Simulated Annealing. Table 2.3 shows the information of Toronto dataset.

**Table 2.3** Toronto Dataset (Qu et al., 2009)

Problem Instance	Exams	Students	Enrollments	Conflict Density	Timeslots
car91 I	682	16925	56877	0.13	35
car91 II	682	16925	56242/56877	0.13	35
car92 I	543	18419	55522	0.14	32
car92 II	543	18149	55189/55522	0.14	32
ear83 I	190	1125	8109	0.27	24
ear83 II	189	1108	8014	0.27	24
hec92 I	81	2823	10632	0.42	18
hec92 II	80	2823	10625	0.42	18
kfu93	461	5349	25113	0.03	42
lse91	381	2726	10918	0.06	18
pur93 I	2419	30029	120681	0.03	42
pur93 II	2419	30029	120686/120681	0.03	42
rye92	486	11483	45051	0.07	23
sta83 I	139	611	5751	0.14	13
sta83 II	138	549	5689	0.14	13
tre92	261	4360	14901	0.18	23
uta92 I	622	21266	58979	0.13	35
uta92 II	638	21329	59144	0.13	35
ute92	184	2749	11793	0.08	10
yor83 I	181	941	6034	0.29	21
yor83 II	180	919	6012	0.29	21

#### 2.4.2 Dataset of University of Melbourne

The University of Melbourne dataset was first brought to public by Merlot et al., (2003). Melbourne dataset has 2 timeslots for weekday. The timeslots have different capacity which timeslots I has 28 while timeslots II has 31. The dataset also included some time constraints where the exams can only be assigned to certain time period or the exam can only held in certain session only. Melbourne dataset was focus on minimize some conflicts such as exams on the same day. Table 2.4 shows some information of the University of Melbourne examination dataset.

Cote, Wong and Saboun, (2005) and Merlot et al., (2003), investigated the Melbourne dataset using a bi-objective evolutionary algorithm where Tabu Search and Variable Neighborhood Decent were utilized.

**Table 2.4** University of Melbourne datasets

Problem Instance	Exams	Students	Enrollments	Timeslots
I	521	20656	62248	28
II	526	19816	60637	31

### 2.4.3 Dataset of University of Nottingham

Merlot, Boland, Hughes and Stuckey introduced dataset of University of Nottingham at the PATAT conference in 2002. In this dataset, the total number of student assigned for each timeslot cannot be more than the total room capacity. The aim of this dataset is to minimize the students sit exams in a row.

In 1999, Burke and Newall used Graph Heuristic with the aim of avoiding second-order conflicts on the same day. In 2003, the same method was used by Merlot on this dataset. In addition, Burke solved the Nottingham dataset by using Great Deluge Algorithm in 2004. Table 2.5 shows some information of the University of Nottingham examination dataset.

**Table 2.5** University of Nottingham datasets

Exams	Student	Enrollment	Conflicts Destiny	Timeslots	Capacity
800	7896	34265	0.03(3%)	23	1550

### 2.4.4 Dataset of University Kebangsaan Malaysia (UKM)

Ayob et al., (2007) introduced a capacitated timetabling dataset – UKM dataset. The data presented is real data for undergraduate exam in UKM for Semester I, year 2006. The constraints of this dataset are all exams must be scheduled and student should not be allocated into exams at the same time. More than that, this dataset requires some exam to be held in certain room. Next, students should be assigned into the same room if they are having consecutive exams. The main objectives of UKM dataset are to avoid letting student

have consecutive exams in the same day and to evenly spread the exams of each student. Table 2.6 shows the information of the UKM dataset and Table 2.7 shows the room capacity of this dataset.

**Table 2.6** University Kebangsaan Malaysia datasets (UKM06-01)

Exams	Student	Enrollment	Timeslots	Capacity	Number of Exams' days
818	14047	75857	42	1550	15

**Table 2.7** Room capacity of dataset UKM06-01

Room	Room Capacity
DPBestari	850
DGemilang	610
Dewan(DECTAR)	610
LobiUtama(DECTAR)	270
Pseni(DECTAR)	152
LobiA(DECTAR)	70
LobiB(DECTAR)	70

#### 2.4.5 Dataset of University Teknologi MARA (UiTM)

The MARA University of Technology (UiTM) is the largest university in Malaysia. Its total number of students was about 100,000.

The UiTM dataset is a capacitated timetabling problem. It was introduced by Kendall and Hussin (2004). There are some rules of this dataset. First, all the exams must be scheduled. Next, it doesn't allow a timetable that have student sit for more than one exam at the same time. The main objective is to spread the exams evenly and avoid having exams in weekend. Table 2.8 shows the information of the UiTM examination dataset.



**Table 2.8** University Teknologi Malaysia (UiTM) dataset

Exams	Students	Enrollments	Timeslots
2063	84675	357761	40

#### 2.4.6 Second International Timetabling Competition (ITC2007) dataset

The Second International Timetabling Competition (ITC2007) was introduced to attract research on techniques concerned timetabling problems encountered within educational institutions. ITC 2007 is a platform to let researcher apply their algorithm or techniques on conducting a timetable that resemble real world situation.

ITC2007 consists of 3 tracks: 1 examination timetabling and 2 on course timetabling. In our research, we only investigate the examination track.

The constraints in ITC 2007 are very well defined and rather comprehensive. This mean ITC 2007 is more resemble the real world situation as it considers room capacity.

There are a lot of techniques has been applied to solve ITC 2007 such as Iterated Forward Search, Hill Climbing and Great Deluge Algorithm by McCollum et al., (2008) ; Gogos, AleFragis and Houses (2008) from Japan uses a multistage approach that uses GRASP, Simulated Annealing and Mathematical Programming to solve it.

Table 2.9 shows some information of ITC 2007. Table 2.10 shows the hard constraints of ITC 2007 and Table 2.11 shows the soft constraints of ITC 2007.

**Table 2.9** Second International Timetabling Competition dataset.

Instance	Conflict Density	Exams	Students	Periods	Rooms	Period HC	Room HC
Exam-1	5.05	607	7891	54	7	12	0
Exam-2	1.17	870	12743	40	49	12	2
Exam-3	2.62	934	16439	36	48	170	15
Exam-4	15	273	5045	21	1	40	0
Exam-5	0.87	1018	9253	42	3	27	0
Exam-6	6.16	242	7909	16	8	23	0
Exam-7	1.93	1096	14676	80	15	28	0
Exam-8	4.55	598	7718	80	8	20	1
Exam-9	7.48	169	655	25	3	10	0
Exam-10	4.97	214	1577	32	48	58	0
Exam-11	2.62	934	16439	26	40	170	15
Exam-12	18.45	78	1653	12	50	9	7

**Table 2.10** Hard constraints of ITC 2007.

Hard Constraints	
H1	Student cannot sit more than one exam at the same time.
H2	Room capacities are always respected.
H3	The exam length should not violate the timeslot length.
H4	A sequence or ordering of an exams must be respected, e.g. schedule Exam A after Exam B;
H5	Schedule exam into specified room (room related hard constraints) e.g. Exam A should schedule to Room 11.

**Table 2.11** Soft constraints of ITC2007.

Soft Constraints	
S1	<i>Two exams in a row:</i> minimize student sitting consecutive exams in the same day.
S2	<i>Two exams in a day:</i> minimize student sitting more than two exams in a day (only applied if more than two timeslot per day).
S3	<i>Spreading of exams:</i> Each set of student examinations should be spread as evenly as possible over the exam period.
S4	<i>Mixed duration:</i> minimize number of exams with different duration that are scheduled into the same room.
S5	<i>Larger examination schedule late in the timetable:</i> minimize the number of large exams appear 'late' of the timetable.
S6	<i>Period penalty:</i> minimize the number of exams scheduled in period with penalty.
S7	<i>Room penalty:</i> minimize the number of exams scheduled in room with penalty.

### 2.4.7 Summary of datasets

After viewing various dataset, we shall make a comparison and summary of them. Table 2.12 show the summary of various dataset.

**Table 2.12** Summary of datasets

Constraints		Toronto	Nottingham	Melbourne	UKM	ITC2007
Examinations	Clash free	Hard	Hard	Hard	Hard	Hard
	Scheduled all exams	-	Soft	Soft	Hard	
	Exam preference <ul style="list-style-type: none"> <li>- Specified arrangement: <math>sa</math></li> <li>- Specified room: <math>sr</math></li> <li>- Large exam schedule first: <math>lf</math></li> <li>- Restriction on exam in particular timeslot: <math>rt</math></li> <li>- Scheduled combined exam in the same timeslots: <math>ct</math></li> </ul>	-	-	Hard ( $rt$ )	-	Hard( $sa$ ) Soft ( $lf$ )
	Consecutive exam <ul style="list-style-type: none"> <li>- Two exam in a row: <math>2r</math></li> <li>- Two exam in a day: <math>2d</math></li> <li>- Two exam in a row overnight: <math>2n</math></li> <li>- Three exam in a day: <math>3d</math></li> </ul>	-	Soft ( $2d$ & $2n$ )	Soft ( $2d$ & $2n$ )	Hard ( $3d$ ) Soft ( $2r$ )	Soft ( $2r$ and $2d$ )

Constraints		Toronto	Nottingham	Melbourne	UKM	ITC2007
Timeslot Related	Timeslot preference - Minimise/avoid usage <i>tu</i>	-	-	-	-	Soft( <i>tu</i> )
	Timeslot length - Mixed duration of exams in one timeslot: <i>mt</i>	-	-	-	-	Hard Soft( <i>mt</i> )
	Spreading - Specified spread: <i>ss</i>	Soft	Hard ( <i>ss</i> )	Soft	Soft	Soft( <i>ss</i> )
Rooms related	Room distance	-	-	-	-	-
	No sharing of room with other exams - For specified exam only: <i>se</i>	-	-	-	Hard ( <i>se</i> )	-
	Room Preference - Consecutive exam scheduled in the same room: <i>cr</i> - Minimise/ avoid usage: <i>ru</i> - Specified room: <i>sr</i>	-	-	-	Hard	Hard( <i>sr</i> ) Soft ( <i>ru</i> )
	Split exam into different rooms - Same building only: <i>sb</i> - As close as possible: <i>cp</i>	-	-	-	-	-
	Capacity - Total seats: <i>ts</i> - Individual room: <i>ir</i>	-	Hard( <i>ts</i> )	Hard( <i>ts</i> )	Hard( <i>ts and</i> <i>ir</i> )	Hard( <i>ir</i> )

Hard =Hard constraint; Soft =Soft constraint; shaded cell = constraint not considered.

## 2.5 Methodologies applied to the examination timetabling problem

Researchers has been studying on timetabling problem since 1960s. There are a lot of techniques or algorithm that has been used to solve timetabling problem. For example: Greedy Algorithm, Genetic Algorithm, Tabu Search and Particle Swarm Optimization. These techniques/algorithm were developed for solving these examination timetabling problems.

These methodologies allows student to conduct a timetable that based on their preference and even better than what they expected. These methodologies also help in prevent negative influence such as clashing time periods in exam.

### 2.5.1 Hill Climbing (HC)

Hill Climbing also was called as simple decent. It is a classical local search algorithm. Hill Climbing is a technique that will chose one candidate solution random from neighboring solution,  $N(s)$  in its every iteration. The candidate solution is represented as  $s'$ . If the candidate solution is better than current one, it will replace the current solution.

Many researchers have hybridized Hill Climbing with other search algorithm because Hill Climbing has an obvious disadvantage. It could be trapped in the local optima. This mean the iteration could not provide a better solution anymore although there is exist of better solution. By hybridize with others technique or algorithm, this problem can be solved. For example, a hybrid of Hill Climbing and Genetic Algorithm was created by Burke, Newall and Weare (1996) to avoid the problem that mention above.

Figure 2.1 will shows the Hill Climbing algorithm procedure.

**Figure 2.1** The Hill Climbing procedures

```

sinceLastMove:=0
While sinceLastMove<1000000 do
  Choose exam e and period t at random s. t. t!=period(e)
    If penalty (e, t)<= penalty (e, period (e)) then
      Move exam e to period t
      sinceLastMove := 0
    Else
      sinceLastMove += 1
    Endif
Done

```

### 2.5.2 Tabu search (TS)

Tabu Search was proposed by Glover (1986) to solve combinatorial optimization problems. Tabu Search is very similar to Hill Climbing but it fixed the local optima problem.

The basic concept of Tabu Search is an extension of steepest descent by incorporating adaptive memory and responsive exploration. In Tabu Search, it will explore the neighborhood of his current solution. If the neighborhood solution has the lowest value compared to current one, it will be accepted although the value might stand a chance to be worse than current solution. It will explore the area other than local optima by accepting a non-improving move. However, this choosing process will usually will lead to a cycling. Hence, a memory which is Tabu List will be used to store recently selected solution to prevent the search stuck in the local optima. And these moves which stored in Tabu List are not allowed to be performed until certain number of looping. However, there is a mechanism called Aspiration Criterion can make the solution free if the solution is typically better than the current best solution.

### 2.5.3 Simulated Annealing (SA)

Kirkpatrick introduced Simulated Annealing in 1983. SA is an extension of the simple descent algorithm but applies a less strict acceptance rule. SA was inspired by the physical annealing process of heating and cooling.

In the first step of SA, it initial an random solution and SA will keep accepting the better solution, while for the worse solution will only being accepted with a low probability. Thompson and Downslan (1998) said that the quality of final solution is very large affected by the cooling schedule of SA. There are 2 types of cooling : fast cooling and slow cooling. Fast cooling tend to lead the search to converge to a local optima, while slow cooling will enhance the solution's quality but at the same time it increases the searching time.

Thompson and Downslan (1996 and 1998) solved the exam timetabling problem with SA in two phases, Constructive (finding a feasible solution) and Improvement (improving the solution quality) phases. An adaptive cooling schedule was used and the results show that it outperformed a simple geometric cooling approach. Thompson and Downslan further their experiment by implementing different cooling schedules and neighborhood in SA, this resulted a significant improvement to the solution's quality.

Frausto and Alonso (2008) hybridized Simulated Annealing and Tabu Search algorithms to solve the Post Enrolment Course Timetabling (track 2) from ITC2007. They divided the algorithm into 2 phases. The first phase is to generate a usable timetable with SA. Second phase, they were still using SA to search for the solution that is nearest to the most optimal result, within a particular time limit. After the SA shows no improvement after the particular time, Tabu Search will take over the job. This method has been proved to be able to produce a usable timetable.



Zhang et al. (2010) used SA to solve the high school exam timetabling problem. They proposed another neighborhood structure that swaps examinations between pairs of timeslots. With this method, SA become more efficient and the performance has been increased.

#### **2.5.4 Great Deluge Algorithm (GDA)**

Great Deluge Algorithm (GDA) operates in a similar way to SA. GDA is like an alternative to SA. What makes the difference is GDA uses an upper limit (can be called as water level) as the boundary of acceptance, unlike SA uses temperature.

GDA starts with initial a solution that its quality is equal to boundary. It accepts worse solution if the cost (objective value) is less than the boundary which is lowered in each iteration according to predetermined rate (known as the decay rate). The procedures of GDA can be seen at Figure 2.2.

Due to the advantage of using less parameter, GDA has been used favorably in several other implementations of meta-heuristics.

Figure 2.2 show the Great Deluge Algorithm procedure. GDA only involves one parameter setting (decay rate) which is an advantages over SA (among others), since the effectiveness of a meta-heuristic technique is often dependent on parameter tuning (Petrovic and Burke, 2004).

**Figure 2.2** shows the GDA procedures

```

Choose an initial configuration
Choose the "rain speed" UP>0
Choose the initial WATER-LEVEL>0
    Opt: choose a new configuration which is a stochastic small perturbation of the old
configuration
    Compute E:=quality (new configuration)
    If E >WATER_LEVEL then
        Old configuration := new configuration
        WATER_LEVEL := water_level+up
    If a long time no increase in quality or too many iteration
    Then stop
    Goto Opt

```

Burke and Newall (2003) investigated GDA on examination timetabling problems. The decay rate is computed as the initial solution multiplied by a user provided factor divided by the number of iterations. The algorithm was run up to 200,000,000 iterations and the search will be terminated if there was no improvement in the last 1,000,000 iterations. They have made a comparison between GDA, SA and Hill Climbing which GDA has been concluded much more excellent than the other two.

McCollum et al. (2009) used an Extended Great Deluge to solve ITC 2008 examination dataset. His method consists of 2 phases: construction and improvement. McCollum constructed an initial solution with an adaptive ordering heuristic. Then, improvement that includes a reheating mechanism is applied. This method has returned some good solutions.

Abdullah et al. (2009) hybridized GDA with TS. The algorithm applied four neighborhood moves at every iteration and the best generated solution will be selected. In particular time, if there is no improvement of the solution's quality then the boundary is increased randomly within value zero and three. This method shown some good result when it is being used on solving course timetabling problem.

#### **2.5.5 Genetic Algorithms (GA)**

Genetic Algorithms (GA) were popularized by Holland (1975). It is an evolutionary algorithm that mimics the process of natural selection. GA is a population based search which uses the principle of biological evolution to generate better solutions from one generation to another (Ross and Corne, 1995 and Burke et al. 2010a).

The methodology consists of operators that known as genetic operators such as mutation, crossover and selection. These operators will manipulate individual solutions (named chromosome in GA) in a population to improve the cost value. The chromosome is represented as a string that holds the information of solution. When practicing GA, there are a few things that should be taken into consideration such as the population size, mutation rate, crossover rate and amount of generations (Goldberg 1989, Pham and Karaboga 2000, Burke and Kendall 2005).

In GA, it will initial a population of random solutions. Every solution has its own cost value (or fitness value) that evaluated based on an equation. Next, the individual will be processed in the recombination phase where crossover and mutation are used to explore the solution space, thus creating new individuals. The new created individual will replace the old individual that has worst fitness value. The process keeps repeating until a termination criterion is met. The termination criterion is dependent, it could be certain loop number or certain time period. Figure 2.3 shows the procedure of GA.

**Figure 2.3** Genetic Algorithm procedure (Cuupic, 2009)

```

Initialize Population P
  For each sol form P
    Calculate Fitness (sol)
  repeat
    select two parent sol1 and sol2 from P
    child = crossover (sol1, sol2)
    mutate (child)
    calculate Fitness (child)
    replace Some(p, child)
  until stop condition not satisfied.

```

GA is very suitable for solving timetabling problem. However, it has some limitations compared to others method. For example, operating on dynamic data sets with GA is difficult as genomes begin to converge early on towards solutions which may no longer be valid for later data.

### 2.5.6 Ant Colony Optimization (ACO)

The Ant Colony Optimization (ACO) was introduced by Dorigo, Maniezzo and Coloni et al. (1996). ACO is a population based technique and it is stimulate the way ants find their food by laying pheromone on the way.

In ACO, every single ant is used to construct a solution. In the process of searching, all information that gained will act as a pheromone, which it will be used in next stage to generate a new solution.

In 1997, a method named ANTCOL (modified ACO) was introduced by Costa and Hertz. They used ACO and sequential heuristic to address the graph coloring problems. In successive generations, each ant will colors the vertices using dynamic (e.g. saturation degree or recursive largest first) or static (e.g. random, largest first , smallest last) constructive methods. The experiment result had shown that the dynamic method is much better than the static methods and this shows that the ACO can solve the examination timetabling problems successfully.

Dowsland and Thomson (2005) had investigated the application of ACO for the examination timetabling problem. Their main objective is to do a comparison to ANTCOL on typical timetabling graph in term of performance. They also wanted to identify promising constructive heuristic combinations, trail calculations and ANTCOL parameter value. As a result, modified ANTCOL can efficiently minimize the number of timeslot in timetable compared to ACO.

### **2.5.7 Graph heuristics (GH)**

In early days of timetabling research community, Graph Heuristics (GH) played an important role (Carter 1986). In GH, for examination timetabling problems, the exams can be represented by vertices in a graph while the hard constrains between examinations is represented by the edge between the vertices. For the graph coloring problem, each vertices will be assigned different color. Hence there will be no same color for adjacent vertices.

At the beginning, GH is used to solve its own schedule examinations (Carter, 1986). Until recently, researchers hybridize GH with the other technique/algorithm. It can be used to initial a solution and the improvement mechanism will be handled to other technique/algorithm. GH can produce acceptable result in very short execution time. More than that, the implementation of GH is easy. However, GH has different strategies that are used depends on how hard is the problem. Figure 2.4 shows the common strategies that are being used.

**Figure 2.4** Common ordering strategies of graph heuristic

Heuristics	Ordering strategy
Largest degree(LD)	This heuristic will choose the exams that having the most conflict with other exams and schedule it first.
Largest Weight Degree(LWD)	This heuristic is almost similar to the largest degree, the only different is it will choose the exams that have the most number of student conflict and schedule it first.
Largest enrollment (LE)	This heuristic will choose the exams that have the most number of enrollments and schedule it first.
Saturation degree(SD)	This heuristic will choose the exams that have the least number of available timeslots that can be selected and schedule it first.
Random Ordering(RO)	This heuristic will randomly order the examinations.

As a comparison, the Largest Degree (LD) and Saturation Degree (SD) usually produce a better solution (Qu et al., 2009). Carter, Laporte and Lee (1996), used different ordering strategies of GH on exam timetable without considering the conflict and spreading of timetable. The result has shown no significant differences between each strategy in term of performance.

However, some researcher has proved that some hybrid of SD with other strategy will produce some better solution. Burke, Newall and Weare (1998) investigated the effect of random elements in Largest Degree, Color Degree and Saturation Degree using

- (a) Tournament selection -randomly selects one from a subset of the first exams in the ordered list
- (b) Bias selection - selects the first exams from an orders list of subset of all of the exams.

And a good result was obtained from University of Toronto dataset.

### 2.5.8 Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO) was developed by Kennedy and Eberhart in 1995. It is a heuristic global optimization method and also an optimization algorithm which based on swarm intelligence.

In PSO, a bird of a flock is represented as a particle, and the swarm is composed of a group of particles. The position of each particle can be regarded as the Candidate Solution to an optimization problem. Every particle is given a *Fitness Function* designed in correspondence with the corresponding problem. When each particle moves to a new position in the search space, it will remember its personal best ( $P_{best}$ ). In addition to remembering its own information, each particle will also exchange information with the other particles and remember the global best ( $G_{best}$ ). Then, each particle will revise its velocity and direction in accordance with its  $P_{best}$  and the  $G_{best}$  to move toward the optimal value and find the optimal solution.

With the advantages of simple and easy application, less parameter setting required. PSO has been applied to establish the optimal timetabling. Figure 2.5 shows the procedure of PSO.

**Figure 2.5** Procedure of PSO

---

**Original particle swarm optimization**


---

**Step (0) Initialization**

Randomly initialize the positions of all particles  $X = (X_1, X_2, \dots, X_{ps})$  of size  $ps$

Initialize the velocity  $(V_1, V_2, \dots, V_{ps})$

Set generation  $t = 0$

Evaluate the fitness values  $F = (f_1, f_2, \dots, f_{ps})$  of  $X$

Set  $X$  to be  $pbest = (pbest_1, \dots, pbest_{NP})$  for each particle

Set the particle with best fitness to be  $gbest$

**Step (1) Reproduction and updating loop**

for  $i = 1:ps$

Update the velocity  $v_i$  of particle  $x_i$  using Eq. (2)

$$V_i^d \leftarrow V_i^d + c_1 \times rand_i^d \times (pbest - X_i^d) + C_2 \times rand_i^d \times (gbest_i^d - X_i^d)$$

Update the position of particle  $x_i$  using Eq. (3)

$$X_i^d \leftarrow X_i^d + V_i^d$$

Evaluate the fitness value  $f_i$  of the new particle  $X_i$

if  $X_i$  is better than  $pbest_i$

Set  $X_i$  to be  $pbest_i$

end if

if  $X_i$  is better than  $gbest$

Set  $X_i$  to be  $gbest$

end if

end for

Set  $t = t + 1$

**Step (2) If termination condition is not met, goto Step(1), otherwise end PSO**

---



## 2.6 Conclusion

This chapter had introduced the general educational timetabling problem mainly focus on examination timetabling. From the literature, different examination from different institution might have different constraints that need to be fulfilled. Among the dataset, the Toronto, Melbourne and Nottingham dataset had got the most attention from the researchers. There are also other dataset such as the UKM, UiTM and ITC2007 which are gaining popularity.

Various methodologies (e.g. heuristic, meta-heuristic and hyper heuristics) especially meta-heuristic had been applied to solve the benchmark examination timetabling problem. However, the success of meta-heuristics is dependent on the parameter tuning (Petrovic and Burke, 2004) which would be a problem for non-experts (e.g. a timetable officer).

In this research, we propose Particle Swarm Optimization as the method to solve examination timetabling problem. The dataset that we will work on is ITC2007. The reason of choosing PSO was because it is a very simple method compared with the other algorithms and it is easily completed as it needs fewer parameters.

Further discussions on PSO will be on the following chapter.

## **CHAPTER 3**

### **METHODOLOGY**

#### **3.1 Introduction**

For recent year, researcher has been used various techniques such as Tabu Search, Great Deluge Algorithm to solve exam timetabling problem. As single based techniques, Hill Climbing, GDA, Tabu Search and SA are very well-known and have been widely used by researchers. While for population based techniques, there are Variable Neighborhood Search, Ant Colony Optimization and GA.

Among these algorithms, PSO has been proven that it can achieve excellent result. PSO requires lesser parameter compared to other algorithm and it fits dynamic environment, this makes PSO easier to implement compared to other algorithm.

PSO will be my choice for solving the timetabling.

#### **3.2 ITC 2007 Examination Track**

ITC 2007 Examination Track is the dataset that I will solve using PSO. Several techniques even combinational of these techniques have been used to solve this examination track.

For example, the winner of the competition of ITC 2007 for Examination Track, Tomas Muller combined Iterative Forward Search (IFS) algorithm, Hill Climbing (HC)

algorithm and Great Deluge (GD) algorithm. He used IFS to find a complete solution. Then, HC will help him to optimize the solution until it cannot be optimized anymore. Lastly, an altered GDA that allows some oscillation is used to generate best solution where HC cannot reach.

The second place winner of ITC 2007 Examination Track was Christos Gogos from Greece. He used Greedy Randomized Adaptive Search Procedure (GRASP) method with the combination of other metaheuristics method. In his method, there are 3 stages before getting the solution. First stage is construction of a high quality feasible solution. This solution will be improved through second stage using Simulated Annealing (SA) local search. Lastly, a mathematical programming will further enhance the quality of the solution.

The third place winner, Atsuta's group from Japan formulates the ITC 2007 as an instance of Constraint Satisfaction Problem (CSP). Then, they apply a powerful CSP solver which adopted Tabu Search and Iterated Local Search to find the solution.

From the statements above, we can tell the method for solving exam timetabling are various and all of these technique could achieve high performance where their scores (duration of finding solution, the shorter the better) are relatively close.

### 3.3 Problem Formulation

The ITC 2007 examination timetabling problem model will be explained here.

Indices

$I_j$       $1 \dots N$

$R_p$       $1 \dots R$

$S$         $1 \dots S$

$T$         $1 \dots T$

### Parameters

N	The number of examinations
R	The number of examination rooms
S	The number of students
T	The number of available timeslots
$S_i$	The number of registered student in exam i
$R_t$	The number of examination rooms available at timeslot t
$B_r$	The building for room r
$f_r$	The total capacity for room r
$c_{ij}$	The conflict matrix where each element $(c_{ij}, i, j \in \{1 \dots N\})$ is the number of students that have to take both exam i and j. The conflict matrix is a symmetrical matrix of size N, where diagonal elements $c_{ij}=S_i$

### Decision variables

$x_{ir}$	1 if examination i is assigned to timeslot t, 0 otherwise
$y_{ir}$	1 if examination i is assigned to room r, 0 otherwise
$z_{rt}$	1 if room r is assigned to timeslot t, 0 otherwise

The objective is to spread out examinations over the exam period (timeslots) for each student, minimise splitting an exam over several rooms. Therefore our formulation is as follows:

$$(\text{Minimise}) F(x) = F_1 + F_2 + F_3 \quad (\text{Eq.1})$$

The first component of the cost,  $F_1$  (spreading the exams over the exam period,  $S_E1$ ) is shown in Eq.2.

$$(\text{Eq.2})$$

$$F_1 = \frac{\sum_{i=1}^N \sum_{j=1}^N c_{ij} \cdot \text{proximity}(t_i, t_j)}{2S}$$

and

Where  $m_i$  is the number of rooms exam  $I$  has been split across. It can be calculated using the following formulation,  $m_i = \sum_{r=1}^R y_{ir} \forall i \in \{1, \dots, N\}$ . Eq.5 represents a cost for an exam  $I$  that us being penalised for splitting the exam in multiple room ( $m_i > 1$ ). For example, if an exam is being split into rooms, then a value of 1 is given as the penalty value. Splitting the exam across 3 rooms corresponds to a penalty of 2 and so on.

Eq.1 is subject to the following constraints:

- a) No student can sit two exams concurrently (clash-free requirement,  $H_{E1}$ ). If examination  $I$  and  $j$  are scheduled in timeslot  $t$ , the number of students sitting both examination  $I$  and  $j$  must be equal to zero, i.e.  $c_{ij}=0$ . This hard constraints is expressed in Eq.6:

(Eq.6)

$$\sum_{i=1}^N \sum_{j=1}^N \sum_{t=1}^T x_{it} x_{jt} c_{ij} = 0$$

- b) All exams must be scheduled and each exam must be scheduled only once in available timeslots,  $T$  (see Eq.7)

(Eq.7)

$$\sum_{t=1}^T x_{it} = 1 \text{ For all } i \in \{1, \dots, N\}$$

- c) Only one examination paper is scheduled to a particular room in a particular timeslot,  $H_{E3}$ . There Is no sharing of rooms with other exam papers (even though seats might be available to fit in another exam), except for requested combined exams, which has been carried out as a pre-process operation (see Eq.8).

(Eq.8)

$$\sum_{i=1}^N x_{it} y_{ir} = z_{rt} \text{ For all } t \in \{1, \dots, T\} \text{ and for all } r \in \{1, \dots, R\}$$

- d) Exam can only be split across several rooms in the same building,  $H_E4$  (see Eq.9).  
(Eq.9)

$$\sum_{r=1}^{R-1} \sum_{p=r+1}^R y_{ir} y_{ip} b_{rp} = \frac{m_i(m_i - 1)}{2} \text{ For all } i \in \{1, \dots, N\}$$

where

$$b_{rp} = \begin{cases} 1 & \text{if } (B_r = B_p) \\ 0 & \text{otherwise} \end{cases}$$

- e) For each timeslot  $t$ , the number of rooms assigned to a particular timeslot must not exceed the maximum number of rooms available in a timeslot,  $R_t$  (see Eq.10)  
(Eq.10)

$$\sum_{r=1}^R z_{rt} \leq R_t \text{ for all } t \in \{1, \dots, T\}$$

- f) The total number of students assigned to a particular exam room(s) must be less than the total room capacity (see Eq.11).  
(Eq.11)

$$S_i \leq \sum_{r=1}^R y_{ir} f_r \text{ For all } i \in \{1, \dots, N\}$$

### 3.4 Particle Swarm Optimization to solve ITC 2007 examination timetabling problem

In this section, details of how do PSO solve ITC 2007 will be discussed.

The proposed PSO algorithm can be applied to create efficient and feasible exam timetable. Below, we will discuss about general idea of PSO.

At the beginning of PSO, the initial velocity and position of each particle in a group of particles are determined randomly. Then, there will be an evolving process as follows:

1. The initial velocity and position of each particle in a group of particles are determined randomly.
2. The fitness value of each particle is calculated according to the defined objective function.
3. If the fitness value of each particle's current location is better than its Pbest , the Pbest is set to the particle's current position.
4. The fitness value of the particle is then compared with that of the Gbest. If it is better, the Gbest is updated.
5. Equation as shown below is applied to update the velocity and position of each particle.
6. Iterate Step 2 until the termination criteria is met or the optimal solution is obtained.

$$V_{id}^{t+1} = V_{id}^t + c_1 \times Rand1 \times (P_{id} - X_{id}) + c_2 \times Rand2 \times (P_{gd} - X_{id})$$

$$X_{id}^{t+1} = X_{id}^t + V_{id}^{t+1}$$

Where

$V_{id}$  is the velocity component of the  $i$ th particle in the  $d$ th dimension.

$X_{id}$  is the position of the  $i$ th particle in the  $d$ th dimension.

$c_1$  is the cognitive learning factor.

$c_2$  is the social learning factor.

$P_{id}$  is the position component of the Pbest of the  $i$ th particle in the  $d$ th dimension.

$P_{gd}$  is the position component of the Gbest of the  $d$ th dimension.

Rand() is a random number between [0,1].

The cycle will repeat over and over again until an optimal timetable is found or the termination criteria is met.

### 3.5 Conclusion

This research will solve the ITC 2007 exam timetabling problem by using PSO. The ITC 2007 dataset consists of 3 parts of exam track which are Early stage, Late stage and Hidden stage. Before working on it, we will analyse and organize the dataset and simplify it for ease of process in PSO.

By using PSO algorithm, a solution to the ITC 2007 will be generated. There are some steps to obtain the optimal timetable. First, we will initiate the particles randomly. Second, we calculate the fitness value to update Pbest and Gbest regularly. Last, the processs will be repeat until the termination criteria is met.

The generated timetable should not against the hard constraints and soft constraints of ITC 2007. At the same time, we also try our best to shorten the generating time of the timetable.



## **CHAPTER 4**

### **DESIGN & IMPLEMENTATION**

This chapter explains the project development process of scheduling ITC2007 timetable. The process can be generalized into 4 steps: First, read the ITC2007 timetabling information from txt file. Second, schedule initial timetables with graph heuristic method. Third, calculate the penalty value and quality of the timetable. Finally, improve the timetable quality with PSO (Particle Swarm Organization) method.

#### **4.1 Project Implementation**

ITC 2007 consists of 8 dataset while we concentrate on dataset 1, 2 and 4. Each dataset consists of some txt file which stored the information of ITC2007. For example, room.txt, roomHC.txt, exam.txt.

These txt file provides us information of number of students, number of exams, room capacity, timeslot and some other information about ITC2007.

##### **4.1.1 Display the information of the ITC2007 dataset**

Each dataset of ITC2007 consists of 7 txt file. The details on each txt file will be explained as follow.

```

1 180, 2545, 2548, 2586, 2594, 2595, 2612, 2620, 2656, 2661, 2664, 2667, 2681, 2714, 2748, 2751, 2760, 2763, 2771, 2789, 2
2 180, 434, 648, 726, 730, 809, 2510, 2511, 2515, 2521, 2539
3 180, 35, 339, 1262, 1274, 1310, 1383, 1385, 1393, 1415, 1429, 1444, 1457, 1472, 1479, 1511, 1516, 1542, 1561, 1589, 1627
4 180, 1262, 1274, 1310, 1383, 1385, 1393, 1415, 1429, 1444, 1457, 1472, 1479, 1511, 1516, 1542, 1561, 1589, 1627, 1632, 2
5 180, 235, 1318, 1353, 1417, 1455, 1533, 1655, 1663, 1749, 1785, 1882, 1892, 1903, 1914, 1961, 1963, 1964, 2008, 2061, 20
6 180, 1318, 2507, 2620, 2751, 2906, 2943, 3022, 3053, 3308, 3321, 3399, 3577, 3657, 3730, 3731
7 180, 11, 82, 87, 95, 128, 133, 185, 235, 320, 357, 363, 370, 448, 458, 481, 523, 531, 573, 576, 594, 602, 637, 640, 642,
8 180, 5, 49, 59, 140, 171, 190, 200, 237, 245, 247, 252, 285, 317, 348, 351, 447, 482, 497, 554, 729, 760, 793, 823, 831,
9 180, 5, 35, 49, 59, 140, 171, 190, 200, 237, 245, 247, 252, 285, 317, 339, 348, 351, 447, 482, 497, 554, 729, 760, 793,
10 180, 5, 49, 59, 140, 171, 190, 200, 237, 245, 247, 252, 285, 317, 348, 351, 447, 482, 497, 554, 729, 760, 793, 823, 831,
11 180, 11, 87, 95, 128, 133, 320, 357, 363, 370, 458, 481, 573, 576, 594, 640, 642, 756, 829, 861, 931, 949, 1013, 1098, 1
12 180, 11, 82, 87, 95, 128, 133, 185, 320, 357, 363, 370, 448, 458, 481, 523, 531, 573, 576, 594, 602, 637, 640, 642, 756,
13 180, 87, 247, 531
14 180, 2524, 2525, 2526, 2539, 2547, 2589, 2675, 2691, 2734, 2969, 2979, 3065, 3070, 3126, 3128, 3153, 3176, 3180, 3191, 3
15 180, 2547, 2589, 2675, 2691, 2734, 2969, 2979, 3065, 3070, 3126, 3128, 3153, 3176, 3180, 3191, 3204, 3213, 3240, 3245, 3
16 180, 270, 419, 421, 667, 1263, 1267, 1272, 1275, 1278, 1280, 1281, 1282, 1283, 1285, 1286, 1287, 1288, 1292, 1294, 1297,
17 180, 1265, 1315, 1364, 1369, 1443, 1494, 1503, 1513, 1605, 1614, 1712, 1733, 1784, 1821, 1847, 1852, 1889, 1896, 1907, 2
18 180, 395, 759, 1262, 1264, 1268, 1269, 1270, 1276, 1277, 1279, 1284, 1289, 1296, 1300, 1314, 1318, 1322, 1323, 1326, 132
19 180, 79, 190, 200, 255, 285, 320, 348, 351, 394, 447, 497, 567, 576, 642, 651, 712, 749, 760, 767, 854, 875, 876, 967, 9
20 180, 1129, 1265, 1315, 1364, 1369, 1443, 1494, 1503, 1513, 1605, 1614, 1712, 1733, 1784, 1821, 1847, 1852, 1889, 1896, 3
21 180, 69, 77, 89, 148, 334, 358, 397, 578, 677, 693, 752, 1080, 1093, 1211, 1218, 2108, 2301
22 180, 77, 148, 334, 362, 397, 438, 479, 578, 677, 688, 693, 925, 932, 965, 1093, 1103, 1112, 1185, 1190, 1199, 1218
23 180, 532, 1112, 1178, 2301, 2522

```

Figure 4.1 exam.txt

In exam.txt file, the first number in each line shows the duration of exam in minutes. The numbers after that are the student's ID whom is taking that particular exam.

```

1 424, 0
2 219, 0
3 120, 0
4 100, 0
5 40, 10
6 60, 10
7 60, 0
8 40, 0
9 36, 0
10 30, 0
11 30, 0
12 25, 0
13 72, 0
14 40, 0
15 35, 20
16 40, 0
17 38, 0

```

Figure 4.2 room.txt

Room.txt shows the information of room. In the Figure 4.2, there are 17 rooms in total. The first number of each line show the capacity of the room. The second number of each line represent the penalty value of using this room.

1	10:12:2005, 07:55:00, 180, 0
2	10:12:2005, 13:30:00, 180, 0
3	10:12:2005, 19:30:00, 180, 0
4	12:12:2005, 07:55:00, 180, 0
5	12:12:2005, 13:30:00, 180, 0
6	12:12:2005, 19:30:00, 180, 50
7	13:12:2005, 07:55:00, 180, 0
8	13:12:2005, 13:30:00, 180, 0
9	13:12:2005, 19:30:00, 180, 0
10	14:12:2005, 07:55:00, 180, 0
11	14:12:2005, 13:30:00, 180, 0
12	14:12:2005, 19:30:00, 180, 0
13	15:12:2005, 07:55:00, 180, 0
14	15:12:2005, 13:30:00, 180, 0
15	15:12:2005, 19:30:00, 180, 0
16	16:12:2005, 07:55:00, 180, 0
17	16:12:2005, 13:30:00, 180, 0
18	16:12:2005, 19:30:00, 180, 200
19	17:12:2005, 07:55:00, 180, 0
20	17:12:2005, 13:30:00, 180, 0
21	17:12:2005, 19:30:00, 180, 500

Figure 4.3 period.txt

The available timeslot is stored in period.txt. The first data is the date of the exam, the second data is the starting time of the exam. Third data is duration of the exam in minute and fourth is the penalty value.

1	306, EXAM_COINCIDENCE, 307
2	410, EXAM_COINCIDENCE, 419
3	411, EXAM_COINCIDENCE, 418
4	653, EXAM_COINCIDENCE, 656
5	560, AFTER, 300
6	650, AFTER, 296
7	801, AFTER, 649
8	856, EXCLUSION, 246

Figure 4.4 periodHC.txt

PeriodHC.txt stored the period hard constraints of scheduling timetable.





```

1  TWOINAROW, 9
2  TWOINADAY, 5
3  PERIODSPREAD, 2
4  NONMIXEDDURATIONS, 10
5  FRONTLOAD, 50, 10, 5

```

*Figure 4.7 institutionalWeight.txt*

The penalty value of soft constraints are all shown in *institutionalWeight.txt*. TWOINAROW means when two consecutive exams is scheduled, 9 will be counted as penalty and added into quality measurement. TWOINADAY penalty is applied when student take more than one exam in the same day. PERIODSPREAD is applied if the exams taken by a student were not spread as far as possible. NONMIXEDDURATIONS penalty is applied when different exam with different duration assigned into same timeslot. Finally, FRONTLOAD consists of 3 numeric data. First number is the largest number of exam, second number represents the number of last periods and last number represents penalty value.

#### 4.1.2 Coding

The programming language chosen for develop the project is Java. The Netbeans IDE is the tool of development. Before manipulating the data of ITC2007, all the information of txt file is read and stored for usage later.

```

public class LoadMultipleTxt {

    public static void main(String[] args) throws IOException {
        String target_dir = "C:\\Users\\qx\\Desktop\\IIC2007\\dataset1";
        File dir = new File(target_dir);
        File[] files = dir.listFiles();

        for (File f : files) {
            if(f.isFile()) {
                BufferedReader inputStream = null;

                try {
                    inputStream = new BufferedReader(
                        new FileReader(f));

                    String line;
                    int count = 1;

```

*Figure 4.8 Coding for txt file loading*

#### 4.1.2.1 Calculate the information

Before scheduling the timetable, some essential variable have to be calculated. These variable includes “total number of exams”, “total number of student enrollment”, “total number of timeslot” and “total number of room”.

```

Dataset 1 has been loaded :
Total number of exam : 607
Total number of enrollment : 32380
Total number of timeslot : 54
Total number of room : 7

```

*Figure 4.9 the information of dataset 1 after calculated*

```

System.out.println("Dataset " + dataset + " has been loaded :");
System.out.println("Total number of exam : " + numOfExam);
System.out.println("Total number of enrollment : " + numOfEnrollment);
System.out.println("Total number of timeslot : " + numOfTimeslot);
System.out.println("Total number of room : " + numOfRoom);

```

*Figure 4.10 coding of displaying the dataset information*

When the txt files were loading, calculation is done at the same time and the value will be stored into variable such as 'numOfExam', 'numOfEnrollment' and others.

#### **4.1.2.2 Calculate conflict matrix, matrix density, number of exam conflicts and student conflicts**

The conflict density represent the difficulty of scheduling a timetable. The higher the conflict density the more difficult to schedule the timetable.

Number of conflict exam and number of conflict student is calculated before calculating conflict density.

#### **4.1.2.3 Sorting Exams with LE, LWD and LD**

There are 4 sorting method to schedule the timetable which are Largest Enrollment (LE), Largest Degree (LD), Largest Weight Degree (LWD) and random.

LE is to sort the exam according to the number of student that take the exam. This mean the exam with highest number of student will assigned into certain timeslot first, then exam with second highest and follows.

LD sorts the exam according to exam conflict. This mean the exam that most conflict against other exam will be assigned first.

LWD sorts the exam according to the number of student who involved in conflict.

Exam : 103 , LE : 259
Exam : 167 , LE : 258
Exam : 437 , LE : 256
Exam : 443 , LE : 255
Exam : 131 , LE : 253
Exam : 1 , LE : 251
Exam : 32 , LE : 240
Exam : 198 , LE : 237
Exam : 569 , LE : 228
Exam : 447 , LE : 226
Exam : 6 , LE : 219

*Figure 4.11 Exam sorted with ascending LE*

In figure 4.11, exams were sorted according to number of student that has enrolled. The LE were listed in ascending order.

Exam : 426 , LD : 108
Exam : 198 , LD : 99
Exam : 6 , LD : 96
Exam : 221 , LD : 95
Exam : 299 , LD : 94
Exam : 427 , LD : 93
Exam : 425 , LD : 91
Exam : 303 , LD : 90
Exam : 8 , LD : 89
Exam : 300 , LD : 88
Exam : 199 , LD : 86

*Figure 4.12 Exam sorted with ascending LD*



In figure 4.12, exams were sorted according to number of exam conflict with another exam. The LD were listed in ascending order.

Exam : 437 , LWD : 1523
Exam : 6 , LWD : 1396
Exam : 447 , LWD : 1386
Exam : 198 , LWD : 1240
Exam : 5 , LWD : 1094
Exam : 4 , LWD : 1094
Exam : 443 , LWD : 901
Exam : 533 , LWD : 899
Exam : 3 , LWD : 881
Exam : 531 , LWD : 881
Exam : 535 , LWD : 851

*Figure 4.13 Exam sorted with ascending LWD*

In figure 4.13, exams were sorted according to number of conflict student. The LWD were listed in ascending order.

#### **4.1.2.4 Sorting method**

The method used for sorting LE, LD and LWD were quicksort algorithm. It is an efficient and fast algorithm. Below is sample coding of QuickSort function.

```

* @author qx
*/
public class Quicksort {
    private int[] numbers;
    private int number;

    public void sort(int[] values) {
        // check for empty or null array
        if (values == null || values.length == 0) {
            return;
        }
        this.numbers = values;
        number = values.length;
        quicksort(0, number - 1);
    }

    private void quicksort(int low, int high) {
        int i = low, j = high;
    }
}

```

Figure 4.14 QuickSort class

```

[-] public void setUp() throws Exception {
    numbers = new int[SIZE];
    Random generator = new Random();
    for (int i = 0; i < numbers.length; i++) {
        numbers[i] = generator.nextInt(MAX);
    }
}

[-] public void SortLD() {
    Quicksort sorter = new Quicksort();
    sorter.sort(LD);
}

```

Figure 4.15 Calling quicksort function to sort LD

## 4.2 Solving ITC 2007

In my research, we will schedule an initial solution (timetable) using graph heuristic method. The timetable must fulfil all the hard constraints and soft constraints of ITC 2007. However, the quality of the timetable that generated by graph heuristic is quite low. Hence, PSO will be used as a method of improving the quality of that timetable.

### 4.2.1 Initialization of timetable with graph heuristic

The sequence of exam to be inserted into timetable in graph heuristic can be divided into 4 types: largest enrolment (LE), largest degree (LD), largest weight degree (LWD) and random. The explanations of each types has already mentioned in previous section.

First, we should inserts all the exams that has hard constraints. This is because if we inserts those exams after majority normal exam, there is a high chance that all the timeslot left were not suitable. Then, we will inserts other normal exams into timetable according to the sorting method until every exam is fitted into the timetable.

In this progress, we will choose the random timeslot (combination of room and period) for each exam. If the room capacity is fitted, we will have that exam located in that room. Then, we check if the period will cause any clashing of any other exams. Only if these 2 conditions are satisfied, the exam is assigned to that particular timeslot. Otherwise, we have choose another timeslot until suitable timeslot is found.

Not to forget ITC 2007 allows more than one different exams run in the same timeslot. Hence, we also have to check if the sum of all exams in timeslot exceed the room capacity.

The progress continues until all the exams were fitted into timetable. Hence, the initial solution (timetable) is generated.

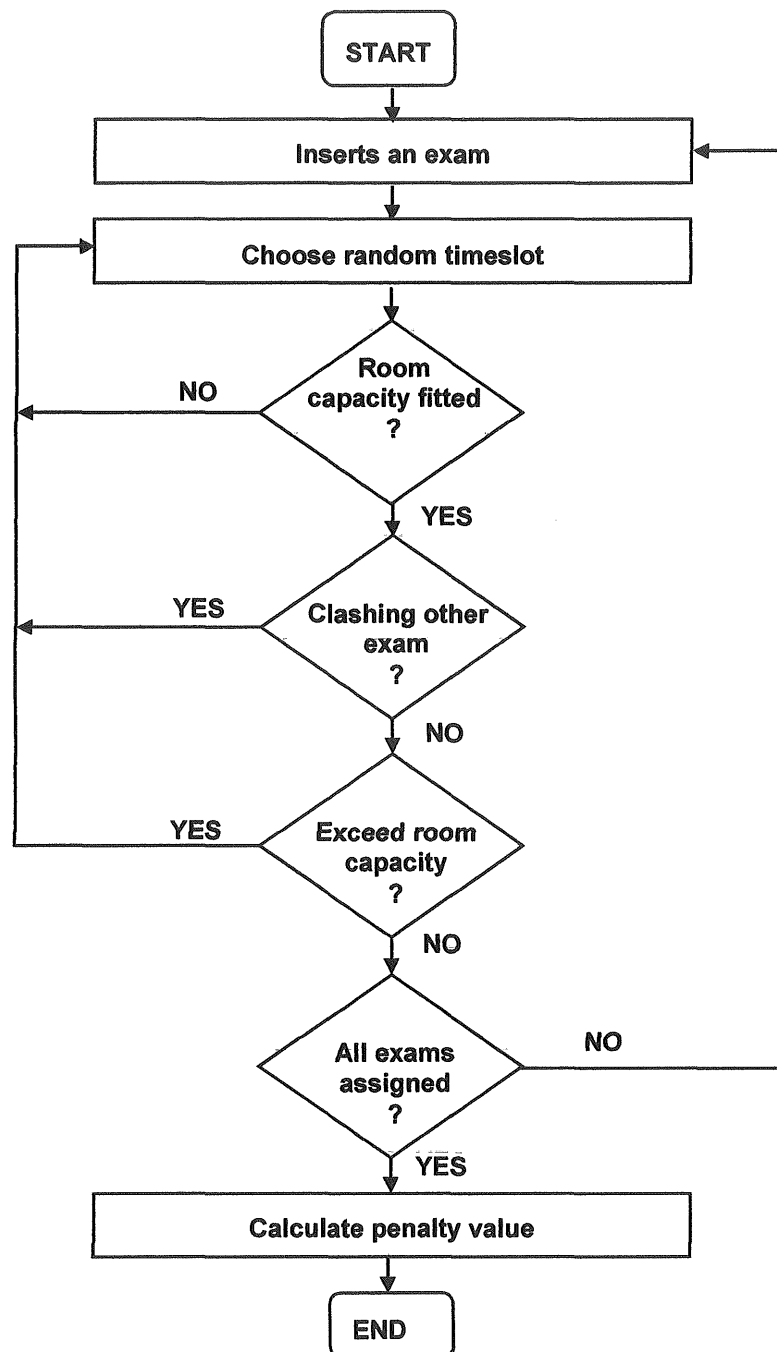


Figure 4.16 Flowchart of generating an ITC 2007 timetable by using graph heuristic.

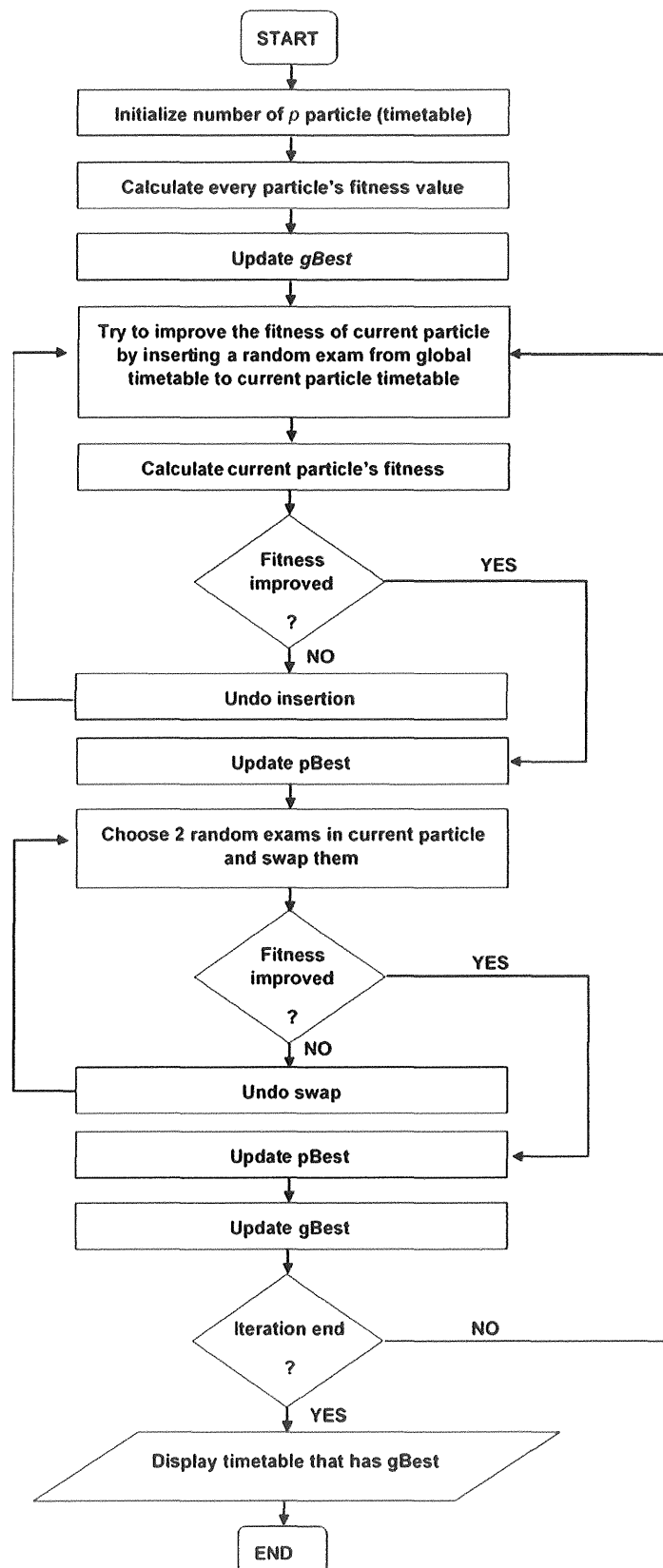
After the timetable has been generated, we will calculate the penalty value of the timetable. The penalty value is determined by the violation of soft constraints. The more number of soft constraints that haven't fulfilled, the higher the penalty value.

There are 7 types of soft constraints in ITC 2007. Every soft constraints has its own institution weight for penalty value calculation. For example, "Two exams in a row" is to count the occurrences where two exams were taken by same student appears after one another. If the number of occurrences was 10 and the institution weight for "Two exam in a row" is 7, then penalty value for this soft constraints is  $10 \times 7 = 70$ . The sum up of 7 soft constraints penalty value is the overall penalty value for that timetable. Every institution weight value is depends on the dataset ITC 2007.

Due to timeslot was chosen randomly, the penalty value will be different for every generating of timetable by graph heuristic.

#### **4.2.2 Improving the quality of initial solution by PSO**

The initial solution timetable will have a high penalty value for sure. This mean the timetable is feasible but the quality is not good enough. We should pursue a better timetable. Hence, PSO is used to improve the quality of the timetable. Figure on next page is the flowchart of PSO and we will have explanation as follows.



*Figure 4.17 Flowchart of timetable quality improvement by using PSO*

PSO is a population technique that requires many particles (timetable) in order to execute algorithm. Therefore, we should decide the number of particle ( $p$ ) first. For example, if we set  $p = 10$  then we should have graph heuristic generates us 10 timetables. Notice that higher number of  $p$  will helps produce better timetable but costs more processing time.

Next, we will calculate the penalty value ( $pBest$ ) for each particle. This mean we will have 10 different penalty value. The timetable that has lowest value (current best timetable) will become global particle and its penalty value will be  $gBest$ . The use of  $gBest$  / global particle is as model of other particle. All the other particles will mimics global particle in order to become better timetable.

So, we copy the pattern of global particle to current particle by inserting random exam from global particle at the same timeslot. For example, global particle has an exam 356 at room 6<sup>th</sup> period 15<sup>th</sup>. We will try to insert exam 356 at room 6<sup>th</sup> period 15<sup>th</sup> at current particle, and the original exam 356 at current particle will be deleted. Of course, the chosen exam and timeslot shouldn't cause any hard constraints violation and clashing problem. Otherwise, we will re-choose another exam and timeslot.

Then, we should take count of the new penalty value for current particle. If the value decreased (improved), we will update the particle and its  $pBest$ . If the value increased, we will undo the insertion and re-choose another exam and timeslot. In other word, the process keep repeating until suitable exam assigned into suitable timeslot and the quality of timetable enhanced.

After the insertion, we will perform 'swap'. Just as its name implies, swap will choose 2 random exam and swap their timeslot. Same as insertion, if the timeslot will causes any hard constraints violation or clashing problem the 'swap' will not be implemented until suitable 2 exam and their timeslot is found. After swap, if the quality of timetable is enhanced we will update the  $pBest$  once again. If the quality dropped we will undo the swap and test another 2 exams until the swap can actually improve the quality of timetable.

Notice that we have 10 particles (as mentioned in example). The ‘insertion’ and ‘swap’ will be executed on every particle. After every particle has been inserted and swapped, we should update the *gBest* because the *pBest* might have surpassed *gBest*. Therefore again, we will have the highest *pBest* set as new *gBest*.

Every update on *gBest* is counted as one generation (iteration). The number of generation is dependent. The higher the number of generation will produce better timetable but it costs more execution time. After certain number of iteration, the execution of PSO will stopped and the improved best timetable shall be chosen.

### 4.3 Conclusion

This chapter discusses the design and the implementation of the research. First, we discussed the ITC 2007 and the information contained. Next, we discussed usage of graph heuristic for generating an initial solution (timetable). Finally, we discussed the improvement of timetable quality by using Particle Swarm Optimization.



## **CHAPTER 5**

### **RESULT & DISCUSSION**

In this chapter, we will discuss on the penalty value from different exam sorting method. We will also evaluate on the performance of Particle Swarm Optimization with different population size.

#### **5.1 Result of penalty value from different exam sorting method**

As mentioned before, the initial solution was generated by graph heuristic with different exam sorting method which are LE, LD, LWD and random. We will only compare the result of LE and LD since LWD couldn't 100% successfully generate a timetable and random method doesn't have any comparability. Below is the result of 10 times running graph heuristic timetable initiation and their statistical view table.

Running	LE	LD
1	38050	43914
2	37558	47410
3	45198	47020
4	46936	43995
5	46486	39544
6	41020	48806
7	46645	43422
8	46183	43697
9	42181	42425
10	45126	45212
<b>Average</b>	<b>43538</b>	<b>44544</b>

*Table 5.1 Result of penalty value from different sorting method*

	Average	Stdev	Var	Min	Max
<b>Largest Enrollment (LE)</b>	43538	3593.5783	12913804.6778	37558	46645
<b>Largest Degree (LD)</b>	44544	2683.902	7235572.5	39544	48806

*Table 5.2 Statistical view of LE and LD*

Stdev = Standard Deviation, Var = Variance, Min = Minimum, Max = Maximum

From the above result, we can tell LE is slightly better than LD in term of producing higher quality timetable. This is because LE could've produce timetable that has penalty value below 40000 while the upper range is within 47000. LD has produced a result 39544 once but according to its Stdev, that was an abnormal result.

However, the advantage of LE compared to LD is very little.

## 5.2 Performance of PSO with different population size

In this section, we analyse the performance of PSO by check its impact to the penalty value deduction. The tested dataset was 2, 5 and 7. The population size test for each set was 2 and 8. Below is the result of performance test and its statistical view.

Dataset	Population size					
	2			8		
	Before	After	Improved	Before	After	Improved
2	24170	9552	<b>60.48%</b>	35954	13094	<b>63.58%</b>
	31160	10400	<b>66.62%</b>	32579	13478	<b>58.63%</b>
	35830	16815	<b>53.07%</b>	28045	11016	<b>60.72%</b>
	35365	21525	<b>39.13%</b>	34786	13963	<b>59.86%</b>
	35730	15886	<b>55.53%</b>	42158	14945	<b>64.55%</b>
5	88517	27846	<b>68.54%</b>	79842	21014	<b>73.68%</b>
	82464	22659	<b>72.52%</b>	84265	24824	<b>70.54%</b>
	76406	21611	<b>71.72%</b>	80456	19591	<b>75.65%</b>
	80694	20344	<b>74.78%</b>	77865	26139	<b>66.43%</b>
	87654	25834	<b>70.52%</b>	86548	24857	<b>71.28%</b>
7	39031	19968	<b>48.84%</b>	48218	25290	<b>47.55%</b>
	46566	27129	<b>41.74%</b>	38951	20340	<b>47.78%</b>
	36795	22322	<b>39.33%</b>	49685	22848	<b>54.01%</b>
	41790	27020	<b>35.34%</b>	37895	19205	<b>49.32%</b>
	44562	24656	<b>44.67%</b>	48952	22581	<b>53.87%</b>

Table 5.3 Result of performance test for PSO

Population size	Average	Stdev	Var	Min	Max
2	56.18867	14.02607	196.73067	35.34	74.78
8	61.16333	9.33206	87.08741	47.55	75.65

Table 5.4 Statistical view of performance test for PSO

Stdev = Standard Deviation, Var = Variance, Min = Minimum, Max = Maximum

From the table above, we found that population size '8' can improved the timetable quality better than population size '2'. The average improvement for population size '2' is only 56% while average improvement for population size '8' is 61%.

However, we also found out that the execution time is longer with '8' at the same iteration number as '2'. Hence, we have conclude the performance of PSO in term of different population size : the higher the population size , the better the quality of timetable but costs more time ; the lower the population size, the execution time required is short but the quality of timetable will be dragged down.

### 5.3 System overview

The implementation of graph heuristic is able to generate a feasible timetable but the quality is so bad. Therefore we have Particle Swarm Optimization to improve the quality of that timetable. Testing of the system shown that PSO is capable of improve the quality by 50% to 60%.

The population size of PSO will give huge impact to the performance of PSO improvement. The more population size will increase the improve rate of timetable but it costs the system more time. In opposite, the lesser population size make the system produce improved timetable fast but the quality is not that good.

The input of the system is the number of particle and iteration number. When these variables is decided and system started doing improvement, the system will start doing improvement until the iteration ends.

The output from the system is the display of the improved timetable. More than that, system will create a txt file that contains timetable information. The txt file can be used for validation by uploading it to ITC 2007 website.

## 5.4 Conclusion

In this chapter, we compared the penalty value of timetable that generated by LE and LD. LE is slightly better than LD but advantage doesn't leading so much.

Next, we looked over the performance of the PSO in term of different population size. The higher the population size the better the timetable quality; the lower the population size the faster the system produce timetable.

Last, we discussed the system overview.

## **CHAPTER 6**

### **CONCLUSION**

In this chapter, we will make a conclusion for all the work in this thesis. Section 6.1 describes the overall introduction for this thesis. Section 6.2 is the result analysis and Section 6.3 talks about future work.

#### **6.1 Introduction**

From the study of different timetabling dataset, we found out that different educational institutions have different requirement for their examination timetable. ITC 2007 is an outstanding dataset because it considered the room capacity compared to other dataset. More than that, ITC 2007 is a very comprehensive dataset. It has a lot of constraints such as ‘exam A must be after exam B’, ‘exam C must be in room D’ and all these constraints causes challenge to the researcher.

With the help of graph heuristic technique and Particle Swarm Optimization, we successfully solved this dataset fulfilling all the hard constraints and reduce the soft constraints violation as much as possible.

#### **6.2 Result analysis**

From the result, we can say that the research is successful and we managed to achieve all the objectives of this project, which are:

1. To study on the examination track of the Second International Timetabling Competition (ITC2007).

2. To implement Particle Swarm Optimization method in solving the examination timetabling problem that satisfies all the constraints.
3. To validate and verify the solution produced using Particle Swarm Optimization whether it satisfies all the constraints.

We have studied the information contains in ITC 2007 examination track. We listed the hard constraints and soft constrains of ITC 2007. We read about penalty value calculation of ITC 2007.

Particle Swarm Optimization has been implemented successfully in our system. The implementation didn't cause any hard constraints violation and fulfilled as much soft constraints as possible.

### **6.3 Future Work**

After proving PSO is capable of solving ITC 2007 examination timetabling problem, we will try to use same proposed technique to solve other dataset. More than that, we will also try to modify PSO and try if it can produce way better result than the current one.

## REFERENCE

1. Russell C. Eberhart, Yuhui Shi, James Kennedy, 2001, Swarm Intelligence. Morgan Kaufmann.
2. Tassopoulos, I. and Beligiannis, G. (2012). Using particle swarm optimization to solve effectively the school timetabling problem. *Soft Computing*, 16(7), pp.1229--1252
3. Chu, S. and Fang, H. (1999). Genetic algorithms vs. tabu search in timetable scheduling. pp.492--495.
4. Chu, S., Chen, Y. and Ho, J. (2006). Timetable scheduling using particle swarm optimization. 3, pp.324--327.
5. Cs.nott.ac.uk, (n.d.). Benchmark Exam Timetabling Datasets. [online] Available at: <http://www.cs.nott.ac.uk/~rxq/data.htm> [Accessed 15 Mar. 2014].
6. Cs.qub.ac.uk, (n.d.). International Timetabling Competition. [online] Available at: <http://www.cs.qub.ac.uk/itc2007/> [Accessed 26 Mar. 2014].
7. Kumar, A., Singh, K. and Sharma, N. (n.d.). AUTOMATED TIMETABLE GENERATOR USING PARTICLE SWARM OPTIMIZATION.
8. Poli, R., Kennedy, J. and Blackwell, T. (2007). Particle swarm optimization. *Swarm intelligence*, 1(1), pp.33--57
9. Tassopoulos, I. and Beligiannis, G. (2012). Using particle swarm optimization to solve effectively the school timetabling problem. *Soft Computing*, 16(7), pp.1229--1252.
10. Web.ist.utl.pt, (2010). Artificial Life by example!. [online] Available at: <http://web.ist.utl.pt/gdgp/VA/pso.htm> [Accessed 15 Mar. 2014].
11. Burke, E. and Ross, P. (1996). Practice and theory of automated timetabling. 1st ed. Berlin: Springer.



12. Burke, E., McCollum, B., Meisels, A., Petrovic, S. and Qu, R. (2007). A graph-based hyper-heuristic for educational timetabling problems. *European Journal of Operational Research*, 176(1), pp.177--192.
13. Davis, L. (1987). *Genetic algorithms and simulated annealing*. Pitman.
14. Deb, K., Pratap, A., Agarwal, S. and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *Evolutionary Computation, IEEE Transactions on*, 6(2), pp.182--197.
15. Dorigo, M. and Stützle, T. (2003). *The ant colony optimization metaheuristic: Algorithms, applications, and advances*. Springer, pp.250--285.
16. Dorigo, M. (2007). Ant colony optimization. *Scholarpedia*, 2(3), p.1461.
17. Glover, F. and Laguna, M. (1997). *Tabu search*. 1st ed. Boston: Kluwer Academic Publishers.
18. Houck, C., Joines, J. and Kay, M. (1995). A genetic algorithm for function optimization: a Matlab implementation. NCSU-IE TR, 95(09).
19. Ingber, L. (1996). *Adaptive simulated annealing (ASA): Lessons learned*.
20. Mahanti, A. and Bagchi, A. (1985). AND/OR graph heuristic search methods. *Journal of the ACM (JACM)*, 32(1), pp.28--51.
21. Mitchell, M., Holl, Forrest, S. and others, (1993). When will a genetic algorithm outperform hill climbing?. pp.51--58.
22. Nahas, N., Khatab, A., Ait-Kadi, D. and Noureldath, M. (2008). Extended great deluge algorithm for the imperfect preventive maintenance optimization of multi-state systems. *Reliability Engineering & System Safety*, 93(11), pp.1658--1672.
23. Noah, S., Abdullah, A., Arshad, H., Abu Bakar, A., Othman, Z., Sahran, S., Omar, N. and Othman, Z. (n.d.). *Soft computing applications and intelligent systems*. 1st ed.

24. Ogata, H., Fujibuchi, W., Goto, S. and Kanehisa, M. (2000). A heuristic graph comparison algorithm and its application to detect functionally related enzyme clusters. *Nucleic acids research*, 28(20), pp.4021--4028.
25. Phys.org, (2014). From chaos to order: How ants optimize food search. [online] Available at: <http://phys.org/news/2014-05-chaos-ants-optimize-food.html> [Accessed 13 Apr. 2014].
26. Pillay, N. (2012). Hyper-heuristics for educational timetabling. pp.316--340.
27. Renders, J. and Bersini, H. (1994). Hybridizing genetic algorithms with hill-climbing methods for global optimization: two possible ways. pp.312--317.
28. Schaerf, A. and Di Gaspero, L. (2001). Local search techniques for educational timetabling problems. pp.13--23.
29. Sivanandam, S. and Deepa, S. (2007). *Introduction to genetic algorithms*. 1st ed. Berlin: Springer.
30. Szu, H. and Hartley, R. (1987). Fast simulated annealing. *Physics letters A*, 122(3), pp.157--162.
31. Talbi, E. (2009). *Metaheuristics*. 1st ed. Hoboken, NJ: Wiley.
32. Unitime.org, (2014). UniTime | University Timetabling. [online] Available at: <http://www.unitime.org/> [Accessed 20 Apr. 2014].
33. Xiao, W. and Dunford, W. (2004). A modified adaptive hill climbing MPPT method for photovoltaic power systems. 3, pp.1957--1963.
34. Yaseen, S. and AL-Slamy, N. (2008). Ant colony optimization. *IJCSNS*, 8(6), p.351.

## APPENDIX

Turnitin Document Viewer - Google Chrome

[https://turnitin.com/dv?o=425366020&u=1028569403&s=&student\\_user=1&lang=en\\_us](https://turnitin.com/dv?o=425366020&u=1028569403&s=&student_user=1&lang=en_us)

PSM1 PSM1 Report - DUE 02 Jun 2014

Originality GradeMark PeerMark

**Final**  
BY EE JUN JIANG

turnitin **28%**  
SIMILAR OUT OF 0

Particle Swarm Optimization to solve the ITC 2007 examination timetabling problem

NAME: EE JUN JIANG

THESIS SUBMITTED IN FULFILLMENT OF THE DEGREE OF COMPUTER  
SCIENCE

No Service Currently Active

PAGE: 1 OF 35