IMPLEMENTATION OF UMP COURSE REGISTRATION SYSTEM USING HEURISTIC METHOD

WONG WEI LEONG

A thesis submitted in partially fulfillment of the requirement for the award of degree of Bachelor of Computer Science (Computer System and Networking)

Faculty of Computer System & Software Engineering
Universiti Malaysia Pahang (UMP)

JUN 2012

ABSTRACT

The Open Registration (OR) system is different from the "Program-based Registration System" which allows students to register their academic subject based on their own study plan. UMP's students are required to make the combination of the study plan to make sure their timetable is suitable with their favorite time. There are some methods on solving the timetabling such as *Tabu Search* (Qu et al., 2009), *Hill Climbing* (Appleby et al., 2011), *Simulated Annealing* (Kirkpatrick and Vecci, 1983) and *Great Deluge Algorithm* (Dueck, 1993) which had been used to solve the College or University timetabling problems. These searching methods fulfilled the automated timetabling system and applied on many systems. These researches will benefit the UMP's student to view the course information with more specific details and convenience the process course registration online.

ABSTRAK

Pendaftaran terbuka (OR) adalah sistem yang berbeza dengan "Sistem Pendaftaran berasaskan Program" yang membenarkan pelajar mendaftar subjek akademik mereka berdasarkan perancangan subjek mereka sendiri. Pelajar UMP perlu membuat gabungan subjek untuk memastikan jadual tersebut sesuai dengan masa kegemaran mereka. Terdapat pelbagai kaedah yang telah digunakan untuk menyelesaikan masalah jadual waktu seperti "Tabu Search" (Qu et al., 2009), "Hill Climbing" (Appleby et al., 2011), "Simulated Annealing" (Kirkpatrick and Vecci, 1983) dan "Great Deluge Algorithm" (Dueck, 1993). Kaedah-kaedah ini telah digunakan dalam sistem untuk mencari waktu subjek yang sesuai secara automatik. Kajian ini dibuat untuk memudahkan pelajar dalam pendaftaran kursus dengan memberikan maklumat kursus yang lebih terpecinci untuk mempercepatkan proses pendaftaran.

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF FIGURES

LIST OF TABLES

LIST OF APPENDICIES

CHAPTER 1

1.0 INTRODUCTION

1.1 Study Background

In Semester 1, 2011/2012, UMP implemented an Open Registration System (OR System) for student's courses registration. Previously UMP implemented "Program-based Registration System" in which the student study plans are pre-determined by the faculty based on the Program Course Structure (i.e. a fixed menu system). The OR System is a flexible menu system that allows students to register their academic subjects based on their own preference (or *study plan*).

According to the UMP Academic Regulation, students can register up to 19 credit hours per semester. Students have to plan their academic load, those prefer minimum weightage could register as low as 12 credits hours, this allows the students to accommodate their learning pace. However, for those who are ambitious they can register up to 19 credits hours per semester. This gives an advantage that allows students to graduate early which can save time and money.

Currently, the course catalog is used to help students plan their preferred courses and timetable. The course catalog is a document that lists all courses offered in that particular semester along with the time and location.

In OR System, almost all of the subjects are offered every semester. However, the capacity of each offered course is normally half of the capacity of the students enrolled. For example, if WAN Technology course (BCN 3203) is a requirement for all Bachelor of Computer Science third year's students with the total 300 students, it will only be offered with the capacity of 180 students in each Semester. Therefore the students can choose to register in either Semester I or Semester II. The overall student's course registration is based on first-register basis (first-come-first-serve basis).

1.2 Problem statement

Currently in the registration systems, students have to manually check for classing and courses availability through the course catalog. The course catalog contains the available courses with the timeslots and rooms. There are many possible combination of the timetable and this confuses the students in selecting the most appropriate timetable. The system does not give any suggestion with the appropriate courses, timeslots and rooms due to having no mathematical model. Therefore, we propose a student course registration system that able to determine the appropriate courses, timeslots and rooms. The objectives and the scopes of the work are presented in the next section.

1.3 Objective

The objectives of this research are:

i. To develop a prototype of UMP courses registration system that allows students to register courses based on the section (timeslots).
ii. To develop a system that able to give suggestion courses on the available timeslots.
iii. To calculate the penalty based on heuristic method concept from the soft constraints as guides for student satisfaction

1.4 Scope

The scope of the work includes:

i. Students are able to register subject and drop the registered subject
ii. Only the courses from faculty FSKKP is available in the registration database

1.5 Research Overview

This research consists of 6 chapters. This chapter represents the study background and the problem statement toward the work. We also stated the objectives as well as the scopes of the work. Additionally we listed the expected research contribution and finally summary of the chapter. The other chapter is organized as following:

Chapter 2 is the literature review for this research. This chapter describes the timetabling overview related to the research problem. It also will discuss the constraints on developing an automated timetabling. Next, the method applied to solve the timetabling problem also discussed in this chapter. The methods included are *Hill Climbing* (HC), *Tabu Search* (TS), *Simulated Annealing* (SA) and *Great Deluge Algorithm*. The UMP course timetabling also have discussed in this chapter.

Chapter 3 describes the methodology used on developing this system. The methodology used in this system is System Development Life Cycle (SDLC). The flows of the system are show in diagram which included Flow Chart, Context Diagram, Data Flow Diagram and Use Case Diagram.

Chapter 4 is the implementation for this research project. The coding of the buttons was explained and the interface will be introduced. The database table used will be displayed.

Chapter 5 is the result and discussion. This chapter presents the system function and the limitation of this system. The further studies stated the ability of this system enhancement.

Chapter 6 is the conclusion of the project.

1.6 Research Contribution

1.6.1 Contribution to the UMP Organization and Community (i.e. student and administrator)

i Implementation of the UMP course registration system for timetabling using heuristic method concept.

1.6.2 Contribution to the scientific community:

i The knowledge gain of the work can be applied on other similar course timetabling problem.

1.7 Summary

In conclusion, we have proposed the project title “Implementation of UMP Course Registration System using Heuristic Method”. We have brief discuss about introduction of this project in this chapter. Firstly, we study the background motivation for this project. Next, we stated the problem statement as well as the objectives for this project. We also fixed the scope for research. The overview of the research will brief describe the work on following chapters. Finally, we stated the research contribution.

Chapter 2

2.0 Literature Review

2.1 Overview of timetabling

University timetabling algorithm is hard to solve for optimality (Malik et al., 2009). University course timetabling problems are considered as NP-hard problem that is difficult to solve to optimality (Ayob and Jaradat, 2009). It is difficult to solve with traditional methods. The computation amount use to find optimal solution increase exponentially due to the big enrollment of students on each year (Turabieh and Abdullah, 2009). The university course timetabling problem involves assigning a set of courses, students and lecturers to a specific number of rooms and timeslots (Abuhamdah and Ayob, 2009) with the weekly assignment of a set of lectures (Shaker and Abdullah, 2009). Tuga et al. (2007) examined course timetabling as the assignment of a set of main academic events related to a course, for example lectures, lab or tutorials sessions, to resources (rooms and timeslots) subject to a set of constraints. Building studying and teaching timetables in large educational institution is not an easy task due to the high number of requirements of lecturers and student attending the classes (Nguyen et al.,

2010). It often takes too much time to do the timetabling by hands and the result timetable may not satisfy the requirements as desired (Nguyen et al., 2010).

Generally, the set of constraint can be categorised as soft and hard (Tuga et al., 2007). Hard constraint is compulsory to fulfill. A timetable will become unacceptable if one of the hard constraints is violated. Soft constraints are some non-compulsory requirements. Soft constraints could be violated but in the minimized of number of violations in order to increase the quality of the timetable. A timetable without any hard constraints violations will be referred to as a feasible timetable (Tuga et al., 2007) in which all courses are assigned to rooms and periods and satisfy all of the hard constraints (Turabieh and Abdullah, 2009).

The constraint can be defined as (Landa and Obit, 2008):

> *"Hard constraints must be satisfied, i.e. a timetable is feasible only if no hard constraint is violated. Soft constraints might be violated but the number of violations had to be minimized in order to increase the quality of the timetable."*

The minimum requirements for feasible timetable, as well as its soft constraints, differ from one university to another, as each university has different rules. However, there might be some common requirements for a timetable to be considered feasible. There are for instance, no students as well as lecturers, are expected to attend two different events at one particular time; a room should not be double-booked. The hard constraints for instance no student is expected to attend two different events at the same timeslots, all events should be assigned to a room within the given timeslots, and the chosen room for an event should meet the specifications required for that event and a room should not be double-booked (Landa and Obit, 2008).

Hard constraints
1) No student can be assigned to more than one course at the same time.
2) The room should satisfy the features required by the course.
3) The number of students attending the course should be less than or equal to the capacity of the room.
4) No more than one course is allowed at a timeslot in each room.

Soft constraints
1) A student should not attend only one course in a day.
2) A student should not attend more than two courses consecutively.
3) A student should not attend a course in the last period in any day.

Figure 2.1: Example soft constraints and hard constraints

(Turabieh and Abdullah, 2009)

2.2 Method applied to solve the timetabling problem

2.2.1 Hill climbing

Hill climbing method is an optimisation algorithm which belongs to the topic of local search which was used for the timetabling method by Appleby et al (1960). It can simple to implement and making it to become a popular first choice. Basically the idea in hill climbing is to always head towards a state which is better that the current one. For example, if a particular are at location *A* and the particular can get to location *B* and location *C* (and the particular target is location *D*) then the particular should make a move *IF* location *B* or *C* appear nearer to location *D* than location *A* does. In steepest ascent hill climbing, it will make the next steps as the best successor of current state, and will only make a move if that successor is better that current state (Rich and Knight, 1991).

1. Start with current-state = initial-state.

2. Until current-state = goal-state OR there is no change in current-state do:

    1. Get the successors of the current state and use the evaluation function to assign a score to each successor.

    2. If one of the successors has a better score than the current-state then set the new current-state to be the successor with the best score

Figure 2.2: Hill Climbing Algorithm (Rich and Knight, 1991)

This algorithm is not attempt to exhaustively to try every path and node, so there is no node list or agenda maintained but just the current state. If there are loops in the search space then using hill climbing shouldn't encounter them but can't keep going up and still get back to the state before. Hill climbing terminates when there is no successor of the current state which is better than the current state itself. Table 2 shows the hill climbing algorithm.

Wang et al. (2008) examined the hill climbing method in centralized job scheduling to determine the migration route. They experiments simulate de-centralized job scheduling, including node adjacencies, local scheduling of grid nodes, and grid workload. It compares with the *k-distributed* and auction methods, hill climbing-based scheduling usually can enhance processor utilization, and can reduce bounded slowdown. Two metric are defined to gauge the efficiently of grid scheduling in the determination of the job migration named as bounded job slowdown of grid and processor utilization of grid. They found that hill climbing-based algorithm can efficiently reduce the bounded slowdown and the grid processor utility.

Nandhini and Kanmani (2009) investigated the implementation of class timetabling using multi agents. They proposed hill-climbing framework (Appleby et al., 1960) for class timetabling to distribute the work involving in, two operation agents (Silva et al., 2003), with different task and objectives have

been introduced. Those are *ComninationGenarator* agents which generate the maximum possible combinations for the input timetable. The one name *MinFinder* agent finds a combination with minimum evaluation function value for further successive examination. By applying the heuristic in the generating combinations, the search space has been reduced and gives the optimal at the earliest. By introducing the agents, *CombinationGenerator* and *MinFinder*, they found that the work has been distributed and the complexity of main task has been reduced.

2.2.2 Tabu search (TS)

Tabu search algorithm is one of the local search algorithms that are popular and applied to a lot of aspects of optimisation problems. According to Qu et al. (2009), the basic procedure of this technique can be divided into two main phases: intensification phase and diversification phase. This technique works by utilizing its components such as *type of tabu list, aspiration criteria, neighbourhood* strategies and etc, with intention to search the best improved solutions. It also makes many researchers who studied on educational timetabling problems interested on it, to use or hybrid with other optimization methods (Malik et al., 2011). *Tabu list* also is an important component of the TS algorithm. The solutions will not be kept in *tabu list* completely, but the accepted move is stored. First in first out (FIFO) rules is used as a data structure for the *tabu list*. Besides that, aspiration criterion is used to override the *tabu restriction* that is the restricted solution can still be selected if it has a lower penalty (cost) that the best solution obtained so far (Al Tarewneh and Ayob, 2011). *Neighbourhood* structure is the most important criteria of a local search used for *simple swap*, which exchanged the rooms assigned and hosting periods to lectures who teaching the different courses. The simple move will applied first

until n consecutive un-improvement iteration trapped (Al Tarewneh and Ayob, 2011).

Nguyen et al. (2010) examined automating a real-world university timetabling algorithm with Tabu Search method. The method starts with building an initial solution using greedy algorithm. The non pre-assigned courses are split into *blocks* and will be assigned to appropriate periods and rooms. However, the combination of available periods, rooms and devices for all courses is a very large number. Next, they applying Tabu Search algorithm to improve the initial solution. In this phase the main kind of moves used is *single moves*. Moreover, two other kinds of moves, including *swap moves* and *block-changing moves*, are used when a number of *consecutive unimproved moves* (unimproved moves are moves that don't change the best solution found so far) have passed. There are three components will keeps in the *tabu list* such as course A_i, block d_i and old period block u_i assigned to d_i. The phase continues by assigning appropriate device into course to complete the solution obtained from previous step. In this phase, they assigned devices into courses due to period assignment and room assignment results taken from previous phases using greedy algorithm. They found that the tabu search-based algorithm that can efficiently applied to the timetabling problem of a university and to other university by modifying the constraints to adapt to new special requirements.

Chu and Fang (2002) compared the Genetic Algorithms (GA) with the TS in timetable scheduling. They examined the timetable-scheduling problem with the constraints related to exam, timeslot, student and lecturer (Turabieh and Abdullah, 2009). They used a simply numbers length list *e* (the number of exams to scheduled), *t* (the number of available timeslots) and each element number between 1 to represent the GA. The interpretation of such a chromosome is that if the *n*th number in the list is *t*, then exam *n* is scheduled to occur at time *t*. For example, the chromosome, [8,11,6,1,2,5,1,2,3,6], is a candidate solution in which exam 1 takes place at time 8, exam 2 takes place at time 11, etc. [3,7,9,2,1,10,4,5,8,11] is another chromosome. Next, they randomly choose a

point, say 3, and crossover them to produce two children as shown in Figure 3. The evolutionary cycle will repeat over and over again until an optimal timetable is found or a certain maximum number of generations are reached. By comparing the results with the TS, they represent the data same as GA approach to encode the timetable. They experiment TS in two basic mechanisms: *tabu restrictions* and *aspiration criteria.* First, they produce a group of 20 candidate solutions from the best solution of iteration. Next, they pick the best candidate solution and mutate it by randomly changing two exams time slots (i.e., two memory values) at a time for ten times to produce the solutions for the next iteration. The best of the new solutions is then selected again to test by *tabu restrictions* and *aspiration criteria.* They may be put into the *tabu list memory* depending on whether they pass the former test. If they fail, but satisfy the latter test, they can also be put in the *tabu list memory.* The advantage of TS over other methods is to use its memorized ability to prevent from searching previous visited areas. Therefore, it is easier to escape from local optimum and approach the global or near global optimum in a short time. Also, the timetables are sorted according to their quality. In this way, they can always pick the best one first to test from the beginning of the list. Finally, they found the TS approach can produce the better timetables than those of GA approach. The search time spent in TS is less than that of GA. GA can produce several different near optimal solution simultaneously.

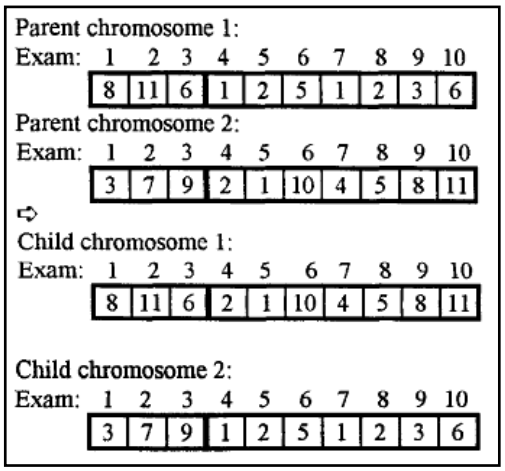

Figure 2.3: Representation and crossover

Turabieh and Abdullah (2009) examined Tabu Search into Memetic approach in solving optimization case that think over on the searching for "highest/lowest point" in the multidimensional parameter space which can be envisioned less or more complex mountainous regions. Memetic method has been applied widely in different fields of optimisation area due to its ability to exploring the search space which greater than local search algorithms or standard evolutionary algorithms. The approach described there is consisting of two phases which is *construction stage* and *improvement phases*. Tabu Search based memetic algorithm solves course timetabling problems with a big set of *neighbourhood* structures are used as a local search mechanism. It begins by creating

```
for i=1 to population size
    Generate random solutions, S_i;
End
Set the length of the tabu, TabuSize;
Choose a best solution from the population, S_best;
Create an empty tabu list with TabuSize, T_List;
Set CanSelect ← True
do while (not termination criterion)
   Select two parents from the population using a roulette
   wheel selection, S_1 and S_2;
   Apply crossover and mutation operators on S_1 and S_2 to
   produce S_1' and S_2';
   if CanSelect== True
      Randomly select a neighbourhood structure, which is
      not in T_List, called Nbs;
   end if
   Apply Nbs on S_1' and S_2' to produce S_1" and S_2";
   Get a minimum solution penalty from S_1', S_2', S_1" and S_2",
   called current solution S*;
   if (S* < S_best)
        S_best ← S*;
        CanSelect ← False;
   Else
        Push Nbs to T_List;
        CanSelect ← True;
   end if
   Update the sorted population by inserting and removing
   good and worse solutions, respectively, while maintaining
   the size of the population;
end while
Output the best solution, S_best;
```

Figure 2.4: Tabu-based memetic algorithm (Turabieh and Abdullah, 2009)