LECTURE TIMETABLING USING HEURISTIC TECHNIQUE

EMILIA BINTI ADZMI

A thesis submitted in fulfilment of the

Requirements for the award of the degree of

Bachelor of Computer Science (Software Engineering)

Faculty of Computer Systems & Software Engineering

Universiti Malaysia Pahang

JUNE, 2012

ABSTRACT

The purpose of this paper is to try to develop a system about lecture timetabling problem. This project presents about Lecture Timetabling using Heuristic Method. Timetabling is a well known difficult combinatorial problem. In recent years, several techniques have been used to automatically generate university timetabling problems, including graph colouring heuristics and a lot more. Timetabling deals with the problem of placing certain resources into a limited number of time slots, subject to given constraints, in order to satisfy a set of stated objectives to the highest possible extent. In this paper, by using the proposed heuristic technique, a lecture timetable is produced considering some constraints that are able to be satisfied. Though it is impossible to consider all constraints in one timetable, the system proposed try to satisfied some of the constraints in order to generate the lecture timetable.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

LTP	Lecture Timetabling Problem
RAD	Rapid Application Development
VNS	Variable Neighbourhood Search
GA	Genetic Algorithm
UMP	Universiti Malaysia Pahang
FSKKP	Fakulti Sistem Komputer dan Kejuruteraan Perisian

LIST OF APPENDIXES

CHAPTER 1

INTRODUCTION

1.1 Background

Due to expand enrolment of students in colleges and university in this recent years, number of students in colleges and universities has increased hugely, some even amount to thousands. Thus it is very difficult to take advantage of surplus resources fully and allocate them effectively without affecting normal activities and constraints. Constructing timetable for lecture timetable is different from constructing a timetable for final examinations. The different between both timetabling is huge due to its great different of constraint conditions in timetabling, so we cannot adopt the same method to solve both of the problems [1].

This system is developed for University Malaysia Pahang (UMP). Lecture Timetabling Using Heuristics put forward heuristics ordering in order to construct the timetable. By using heuristics approach, it allocates lecturer, timetabling period and classroom resources in different phase so as to obtain a feasible solution in order to get a solution satisfying the users.

Heuristic technique is a very versatile technique which can be change depends on the adaptive techniques that are applied in it. Heuristic has been variously presented in the form of proverbs, maxims, hints, suggestions, advice, principle, rules of thumb, criteria, production rules, programs, procedures, option filters and others.

1.2 Problem Statement

The lecture timetabling problem is a common issue to all educational institutions. Lecture timetabling problem are a very difficult issue that are still in argument even today since we need to there are a lot constraints that we need to consider. The examples of constraints are classroom resources, time, lecturer and other constraints. In order to get a feasible solution, it is very important to consider all possible constraints as much as possible. In UMP for example, ever since UMP used open registration for its course registration, it is all the more reason for these constraints to be considered thoroughly.

Depends on institution and constraints, some problem we can see is about the classroom not fit for total student which means maximum total student for subject is bigger than maximum classroom capacity and also lecturer is double booked. So, more research of method should be done so that we can understand more about timetabling problem to aid for a better timetable result.

1.3 Objective

The objectives of this project are:

i. To study the way of scheduling a timetable for the lecture session of FSKKP by using heuristic method.
ii. To design a simple and usable system.
iii. To develop a prototype to schedule the solution for a given case study.

1.4 Project Scopes

The scope of the project is as below:

i. The system is developed for FSKKP of UMP to generate the lecture timetable.
ii. The user of the system is the staff of FSKKP.
iii. The result of the system will be used by students and the lecturers of FSKKP.

1.5 Thesis organization

This thesis consists of 5 chapters. Chapter 1 is the Introduction. This chapter gives a brief explanation on the introduction of system and research. The objectives of the thesis as well as the problem statements, and scope of the project are also stated in this chapter.

Chapter 2 is the Literature Review. This chapter briefly explains the literature review and research for project that has been chosen. The research is divided into two sections which the first one is the case study of an existing systems and the second one is the research for technique that are used to develop the existing systems.

Chapter 3 will discuss on the approach used to develop the proposed system and overall work load to develop this system. The content consists of the approach and framework for the project that used in this system.

Chapter 4 is the implementation. This chapter will explain the process that is involved during development of this system

Chapter 5 is the results, discussion and conclusions. The results that are obtained from the implementation of the system are discussed thoroughly in Chapter five. The constraints of this project are also stated clearly in Chapter five.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This literature review chapter attempts to place an emphasis on the fundamental aspects of the research area. It introduces the definition of the characteristics of university lecture timetabling and specifically the university lecture timetabling problem. Other than that, some reviews of techniques used are also discussed. This chapter also discussed overview of some existing system.

2.2 Background

The university lecture timetabling problem consists in scheduling a set of lectures for each subject within a given number of rooms and time periods. In general, a university lecture timetabling problem consists in finding the exact time allocation within a limited time period (e.g. a week), of a number of events (subject-lectures) and also assign to them a number of resources (a lecturer, a room, etc.) in such a way that a number of constraints are satisfied. The constraints that have to be satisfied by timetable are usually divided into two categories: hard constraints and soft constraints [3].

Hard constraints are those constraints that must be satisfied or rigidly fulfilled. Examples of constrains are, no resource (lecture, room, etc.) may be assigned to different events at the same time, assigned resources to an event (e.g. lecturer) must belong to the set of valid resources for that event (e.g. only specific lecturer can teach a specific subject). The Timetables that meet the hard constraints of the problem are referred to as feasible timetables.

On the other hand, soft constraints are those that are desirable to be fulfilled to the possible extent, but are not fully essential. Soft constraints also can be considered as wish lists of the characteristics that we would like the timetable to possess. Therefore, soft constraints can also be used as optimization objectives for the search algorithm. Some examples of soft constraints are, schedule an event within a particular period (e.g. on evenings), minimize time gaps or travel times between adjacent lectures of the same lecturers.

2.3 Studies of previous work

Lecture timetabling is a well research problem. This section introduced the previous work made regarding the problem of university timetabling including examination timetabling as well as lecture timetabling.

2.3.1 A Study of Heuristic Combinations for Hyper-heuristic Systems for the Uncapacitated Examination Timetabling Problem

Heuristic combinations usually take the form of a list of low-level heuristics that are applied sequentially. This study proposes an alternative representation for heuristic combinations, namely, a hierarchical combination of heuristics. Furthermore, the heuristics in each combination are applied simultaneously rather than sequentially. The study also introduces a new low-level heuristic, namely, highest cost. A set of heuristic combinations of this format have been tested on the 13 Carter benchmarks. The quality of the examination timetables induced using these combinations are comparable to, and in some cases better than, those produced by hyper-heuristic systems combining and applying heuristic combinations sequentially [4].

2.3.2 Extensions to the Heuristic Algorithm for University timetables

This research paper shows the usage of heuristics algorithm for university timetables. It uses a heuristics algorithm in order to solve the timetabling problem in their university. The timetables obtained in this paper show that this algorithm is suitable for Universities and Technical Colleges since every restriction stated in the research have been required [5]. However, the technique applied in the research might not be suitable for all universities and college since not all university or colleges have the same requirements and constrains.

2.3.3 An Informed Genetic Algorithm for the Examination Timetabling Problem

This paper presents the results of a study conducted to investigate the use of genetic algorithms (GAs) as a means of inducing solutions to the examination timetabling problem (ETP). This study differs from previous efforts applying genetic algorithms to this domain in that firstly it takes a two -phased approach to the problem which focuses on producing timetables that meet the hard constraints during the first phase, while improvements are made to these time tables in the second phase so as to reduce the soft constraint costs. Secondly, domain specific knowledge in the form of heuristics is used to guide the evolutionary process. The system was tested on a set of 13 real-world problems, namely, the Carter benchmarks. The performance of the system on the benchmarks is comparable to that of other evolutionary technique s and in some cases the system was found to outperform these techniques. The quality of the examination timetables evolved is within range of the best results produce d in the field [2].

2.3.4 Solving University Timetabling Problems Using Advanced Genetic Algorithms

This paper establishes a new algorithm based on Genetic Algorithms (GA) and sequential local search to solve course timetabling problem. In this research, they perform preliminary experiments on standard benchmark course timetable problems. The GA solution from the research is then compared to the manmade one produced by the institute’s staff and the comparative results are discussed [3].

2.4 Studies of techniques

The induction of examination timetables is a well researched field and various artificial intelligence techniques such as Tabu search, Variable Neighbourhood Search and evolutionary algorithms, including genetic algorithms, have been evaluated for this purpose.

2.4.1 Genetic Algorithm Based Solution

Genetic algorithm is an evolutionary inspired by metaheuristic method that does not guarantee to find optimal solution. There are a number of problems known as deceptive functions that are extremely hard for genetic algorithm. However, in most of cases, genetic algorithm can relatively quickly find a good-enough solution [6].

Genetic algorithm can be said as population-based search method. Genetic algorithms are acknowledged as good solvers for tough problems. However, no standard GA takes constraints into account. This chapter describes how genetic algorithms can be used for solving constraint satisfaction problems [7]. The general scheme of a GA can be given as follows:

```
begin
  INITIALIZE population with random candidate solutions;
  EVALUATE each candidate;
  repeat
    SELECT parents;
    RECOMBINE pairs of parents;
    MUTATE the resulting children;
    EVALUATE children;
    SELECT individuals for the next generation
  until TERMINATION-CONDITION is satisfied
end
```

Figure 2.1: General Scheme of GA

The evaluation function represents a heuristic estimation of solution quality and the search process is driven by the variation and the selection operator. GA has a number of features:

i. GA is population-based
ii. GA uses recombination to mix information of candidate solutions into a new one.
iii. GA is stochastic.

2.4.1.1 Components of genetic algorithm

The most important components in a GA consist of:

i. representation (definition of individuals)
ii. evaluation function (or fitness function)
iii. Population
iv. parent selection mechanism
v. variation operators (crossover and mutation)
vi. survivor selection mechanism (replacement)

2.4.1.1.1 Representation

Objects forming possible solution within original problem context are called *phenotypes*, their encoding, the individuals within the GA, are called *genotypes*. The representation step specifies the mapping from the phenotypes onto a set of genotypes.

Candidate solution, *phenotype* and *individual* are used to denotes points of the space of possible solutions. This space is called *phenotype space*. *Chromosome*, and *individual* can be used for points in the *genotye space*. Elements of a chromosome are called *genes*. A value of a gene is called an *allele*.

2.4.1.1.2 Variation Operators

The role of variation operators is to create new individuals from old ones. Variation operators form the implementation of the elementary steps with the search space.

2.4.1.1.3 Mutation Operator

A unary variation operator is called *mutation*. It is applied to one genotype and delivers a modified *mutant*, the *child* or *offspring* of it. In general, mutation is supposed to cause a random unbiased change. Mutation has a theoretical role: it can guarantee that the space is connected.

2.4.1.1.4 Crossover Operator

A binary variation operator is called *recombination* or *crossover*. This operator merges information from two parent genotypes into one or two offspring genotypes. Similarly to mutation, crossover is a stochastic operator: the choice of what parts of each parent are combined, and the way these parts are combined, depends on random drawings.

The principle behind crossover is simple: by mating two individuals with different but desirable features, we can produce an offspring which combines both of those features.

2.4.1.1.5 Parent Selection Mechanism

The role of *parent selection* (*mating selection*) is to distinguish among individuals based on their quality to allow the better individuals to become parents of the next generation.

Parent selection is *probabilistic*. Thus, high quality individuals get a higher chance to become parents than those with low quality. Nevertheless, low quality individuals are often given a small, but positive chance; otherwise the whole search could become too greedy and get stuck in a local optimum.

2.4.1.1.6 Survivor Selection Mechanism

The role of survivor selection is to distinguish among individuals based on their quality. In GA, the population size is (almost always) constant, thus a choice has to be made on which individuals will be allowed in the next generation. This decision is based on their fitness values, favoring those with higher quality.

As opposed to parent selection which is stochastic, survivor selection is often *deterministic*, for instance, ranking the unified multiset of parents and offspring and selecting the top segment (fitness biased), or selection only from the offspring (age-biased).

2.4.1.1.7 Initialization

Initialization is kept simple in most GA applications. Whether this step is worth the extra computational effort or not is very much depending on the application at hand.

2.4.1.1.8 Termination Condition

Commonly used conditions for terminations are the following:

i. The maximally allowed CPU times elapses
ii. The total number of fitness evaluations reaches a given limit
iii. For a given period of time, the fitness improvement remains under a threshold value
iv. The population diversity drops under a given threshold.

Note: *Premature convergence* is the well-known effect of loosing population diversity too quickly and getting trapped in a local optimum.

2.4.1.1.9 Population

The role of the population is to hold possible solutions. A population is a multiset of genotypes. Most of GA applications have a constant population size which means they doesn’t change during the evolutional search.

2.4.1.2 Genetic Algorithm Basic Structure

A simple genetic algorithm describes the following cycle:

1st Assessment of each individual of the population;
2nd Verification of the termination criteria;
3rd Generation of random n chromosomes that form the initial population;
4th If verify termination criterion - cycle ending;
5th Selection of n/2 pairs of chromosomes for crossover;
6th Reproduction of chromosomes with recombination and mutation;
7th New population of chromosomes called new generation;
8th Go back to step 2.