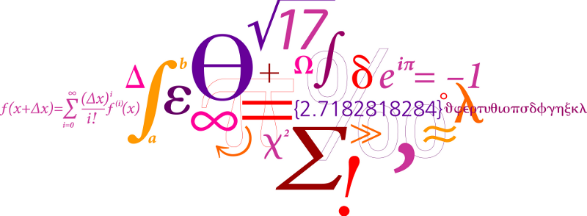


**42137 Optimization using Metaheuristic**

**University Timetabling**

Martin Casper Wiboe

Burak Topal

Too Sheng Tack

**Table of Contents**

Abstract

1. Introduction
2. Problem Description
3. Implementation of Classes
   1. University TimeTabling
      1. Variables
      2. Functions
   2. Heuristic
      1. Variables
      2. Functions
   3. Classes Reading Input
      1. Basic Info
      2. Curriculum
      3. Courses
      4. Lecturers
      5. Rooms
      6. Schedule
      7. Unavailability
   4. XORShiftRandom
4. Metaheuristic
   1. HillClimber
   2. Simulated Annealing
   3. TABU
5. Parameter Tuning
6. Comparison of Methods
7. Conclusion
8. Appendices

**Abstract**

This report will investigate the implementation of Metaheuristic on the university timetabling problem and examine the outcomes. A brief introduction of the problem is initially outlined. Several Metaheuristic methods such as Hill Climber and Simulated Annealing with Taboo Search are applied to minimize and measure the objective value. Parameter tuning is also implemented and the comparison of the results of each method is drawn.

**1.0 Introduction**

Every semester universities face the problem of creating good feasible timetable due to many complex constraints that have to be taken into consideration. Associated penalties will be assigned if the constraint is violated. The purpose of this report is to plan as many lectures as possible and avoid unwanted attributes to minimize the objective which is the penalty. Given a specific time, by implementing Hill Climber and Simulated Annealing we can find the best solution (which may not be the optimum) to the problem. Providing the universities an optimization tool can help them to smooth out the planning and better utilize their resources.

**2.0 Problem Description**

This problem consists of weekly scheduling of the lectures of the courses within a certain number of rooms and time periods, where conflicts between courses are set according to the curricula published by the university. Due to the complexity, the formulation of the problem is simplified and more attributes could be added to enhance the quality of scheduling. In addition to that, this problem consists of several entities, constraints and components of the objective.

**2.1 Entities**

* Days, Periods and Timeslots

Given a fixed number of days per week and each day is split into a certain number of periods. A timeslot is a pair of day and periods.

* Courses and Lectures

Each course that taught by only a lecturer has a required number of lectures to be given and each course will be attended by an associated number of students. Lectures of each course must be spread into a minimum number of working days, moreover there are some periods that a lecture cannot be scheduled in.

* Rooms

Each room has a capacity, expressed in term of available seats.

* Curricula

A curriculum is a group of courses that any pair of the courses have students in common.

The solution of the problem is a number of assignments of lectures to a time slot (day and period) and a room.

**2.2 Constraints**

The following constraints have to be obeyed in order to produce a feasible timetable.

* **Lecture**

A predetermined amount of lectures is assigned to each course. Each lecture must be scheduled in different time slot and the total number of lectures cannot be exceeded.

* **Room Occupancy**

Each room can only accommodate one course in a specific time slot.

* **Conflicts**

Lectures of courses in the same curriculum or taught by the same lecturer must be assigned in distinct time slots.

* **Availabilities**

Some courses cannot be scheduled at specific time slots.

**2.3 Objective**

Several attributes contribute to the objective and each unwanted attribute has an associated penalty value.

**Unscheduled Lectures**

Each course has a lecture that is not scheduled will be penalized 10 points.

**Room Capacity**

For each lecture, the number of students attending the course must be less than or equal to the number of seats in the room. Each student over the capacity will be penalized 1 point.

**Minimum Working Days**

Lectures of each course must be spread into a minimum number of working days. Each day below the minimum working day will be penalized 5 points.

**Curriculum Compactness**

For any given curriculum, a lecture is considered as a secluded lecture if it is not adjacent to any other lecture from the same curriculum within the same day. Each secluded lecture will be penalized 2 points.

**Room Stability**

All lectures of a course should be given in the same room. Each extra room used will be penalized 1 point.

**2.4 Mathematical Model**

The problem is represented by a mathematical model. Sets and parameters are defined as well as variables, constraints and the objective.

**Sets**

C – The set of courses

L – The set of lecturers

R – The set of rooms

Q – The set of curricula

T – The set of time slots. I.e. all pairs of days and periods

D – The set of days

T (d) – The set of time slots that belongs to day d є D

C (q) – The set of courses that belongs to curriculum q є Q

**Parameters**

Lc – The predetermined amount of lecture of each course є C

Cr – The capacity of a room є R

Sc – Number of students attending course c

Mc – Minimum number of working days that the course should be spread to

Fc,t – 1 if the course c є C is available at time slot t є T, otherwise 0

X (c1,c2) – 1 if course c1 є C which is different from course c2 є C ( c1 ≠ c2 ) and conflicting, otherwise 0

V (t1,t2) – 1 if the timeslot t1 and t2 which belongs to the same day and adjacent to each other, otherwise 0

**Decision Variables**

Xc,t,r – 1 if class c є C is allocated to room r є R and timeslot r є R, otherwise 0

**Constraints and Functions**

Each course can at most be assigned one room at a specific timeslot and only if the course is available for the timeslot.

Each room can accommodate at most one course in a given timeslot.

Each course can at most be assigned to a maximum number of lectures.

Conflicting course are not allowed to be scheduled in the same timeslot.

The function Vt,r (x) indicates the amount of capacity that room r є R is exceeded in timeslot t є T.

Function Uc(x) indicates the amount of lectures by which course c є C is scheduled below the specified value Lc.

Function Pc(x) indicates the number of room changes by a course c є C.

Function Wc(x) indicates the number of days that the course is scheduled below the minimum working number of days.

Function Aq,t(x) determines if a curriculum in a timeslot has a secluded lecture.

**Objective**

The objective function is the summation of the penalties.

**Complete Model**

Subject to

**4.0 Implementation of Classes**

9 classes are created for easy maintenance which include University Timetabling, Heuristic, Basic Info, Courses, Curriculum, Lecturers, Rooms, Schedule, and Unavailability. Explanation of variables and functions created within the class will be discussed briefly as below.

**3.1 University Timetabling (need to rewrite a bit)**

The purpose of this class is to read the data from given data files, find the best schedule which minimize the objective and write it into a CSV file for drawing chart and the purpose of parameter tuning.

**3.1.1 Variables**

Firstly, files such as basicFile, coursesFile, lecturersFile, roomsFile, curriculaFile, relationFile, unavailabilityFile are created to keep the information of input files. Then, a variable heuristic of type of Heuristic (self-defined class) is created to record the data from the files mentioned above. Followed by generating an initial schedule of type of Schedule (self-defined class) and searching for the optimum.

**3.1.2 Functions**

There are 2 functions operating in this class, which are the main function and startwithParameters function. In the main function, there are 2 options which the program can choose from to start running, one is solve the problem without benchmarking and another one is solve the problem several times and output the intermediate results. Meanwhile startWithParameters function operates as reading data from the input files.

**3.2 Heuristic**

This class is abstract and must be implemented according to the chosen heuristic. It will perform a search with the input schedule as a starting point and return the most optimal schedule.

**3.2.1 Variables**

Several variables are created such as basicInfo (type of Basic Info), curriculum (type of Curriculum), lecturers (type of Lecturers), courses (type of Courses), unavailability (type of Unavailability), rooms (type of Rooms) to contain the given data. In addition to that, other variables such as timeout (integer), countdownStartTime (long), iterationCount (int), int[] courseAssignmentCount are also created.

**3.2.2 Functions**

Numerous number of methods are created in Heuristic and some important functions are shown in the list below.

1. Public Boolean validateSameLecturerConstraint(Schedule schedule)

* Check that the courses taught by same lecturer are not scheduled in the same timeslot.
* Return true if the constraint is satisfied.

1. Public Boolean validateSameCurriculumConstraint(Schedule schedule)

* Check that the courses grouped in same curriculum are not scheduled in the same timeslot.
* Return true if the constraint is satisfied.

1. Public Boolean validateAvailabilityConstraint(Schedule schedule)

* Check that courses are not scheduled in unavailable timeslot.
* Return true if the constraint is satisfied.

1. Protected int[] getCourseAssignmentCount(Schedule schedule)

* Returns an array containing the number of times of each course has been scheduled

1. Public Boolean validateMaximumScheduleCountConstraint(Schedule schedule)

* Validate that no course has been scheduled more than minimum number of times
* Return true if the constraint is not violated

1. Public Schedule getRandomInitialSolution()

* Generate a random initial schedule

1. Public int evaluationFunction(Schedule schedule)

* Calculate the objective value of the schedule

**3.3 Classes Reading Input**

**3.3.1 Basic Info**

A function, Public void loadFromFile(String file) is created to read the number of courses, rooms, days, period per day, curricula, constraints and lecturers from the input file.

**3.3.2 Curriculum**

The purpose of this class is to keep the relation between the courses and curriculum. If a course belongs to a curriculum, the Boolean array isCourseInCurriculum[][] will return true. A public void loadFromFile(String curriculaFile, String relationFile, int numberOfCourses) function is created to read the data from curriculaFile and relationFile and store assignments as an array of Booleans.

**3.3.3 Courses**  
The purpose of this class is to store the information such as lecturer for each course, number of lectures of each course, minimum working days of each course and number of students of each course.

**3.3.4 Lecturers**  
Integer array, int[] lecturers is created to store the identity of lecturers. A public void loadFromFile(String file) function is created to read the data from a lecturers file.

**3.3.5 Rooms**

Integer array, int[] capacityForRoom is created to store the number of seats in a given room. A public void loadFromFile(String file, int numberOfRooms) function is created to read the data from a Rooms file.

**3.3.6 Schedule**

The purpose of this class is to store the schedule that expressed in term of [day][period][room]. A public void Schedule(int days, int periods, int rooms) function is created and first initialized all assignments to -1, which means that no course is assigned to the timeslot and room. Another method which is public String toString() will print the solution.

**3.3.7 Unavailability**

Boolean array is created such as boolean[][][] courseUnavailable which expressed in term of [day][period][course]. Return true if a course is unavailable in the specific timeslot. A public void loadFromFile(String file, int numberOfDays, int numberOfPeriods, int numberOfCourses) function is created to read the data from a unavailability file.

**3.4 XORShiftRandom**

It is a faster, higher quality replacement for java.util.Random. This class is called when generating initial random solution.