Initial Plan

University Student Timetabling as an Optimization Problem

Author: James Davis (1422447)
Supervisor: Frank C Langbein
Moderator: Alun D Preece

CM3203 - One Semester Individual Project, 40 Credits

Table of Contents

Project Description (1) …………………………………………………………………....1

Project Aims and Objectives (2) ..……………………………………………………………..2
Primary Aims and Objectives (2.1)…………………………………………………………...2
Secondary Aims and Objectives (2.2)………………………………………………………….2
Constraints (2.3)……………………………………………………………………………..2
Testing (2.4) ……………………………………………………………………………………3

Work Plan (3) ………………………………………………………………………………...3
Milestones (3.1)…………………………………………………………………………...3
Weekly Aims and Objectives (3.2)..4
Work Plan Extras (3.3)...5
Deliverables (3.4) …………………………………………………………………………..5
Background Research Aims (3.5) ………………………………………………………...5
Ethical Issues (3.6)..5

1. Project Description

Timetabling is a necessary but time-consuming task undertaken by every education provider around the world. The process involves trying to find the optimal timetable for every student and teacher whilst ensuring that there are no clashes (i.e. a student has to be in two different classes at the time). This can be done by hand (through a trial and error type approach) by trying a version of a timetable and seeing if it would work and have no clashes. This method works for smaller education providers, such as small school, due to the lack of constraint complexity. However, when the same problem is attempted on a larger scale, such as for a university, it becomes far more time consuming due to the increase in the number of constraining variables. This presents a reoccurring problem for larger institutions, forcing them to allocate a significant amount of staff time to solve it every time the need arises for a new timetable to be designed.

For this project, I aim to create a program that produces an optimal (or close to optimal) timetable solution given a full list of hard and soft constraints as input. This program has the intention of being used by a larger sized institution, e.g. the School of Computer Science at Cardiff University, to automatically generate their student and staff timetables.

The program will automate the timetabling process which currently demands a large amount of manpower from every academic term to be done by hand, thus saving the university money and freeing up staff for more useful purposes. All the University staff would be required to do is enter the timetables constraints into the program, and the time-consuming timetable generation will be handled by the program instead of by the staff members.

Constraints I will be referring to are categorised into hard and soft. Hard constraints are the most important and must not be violated at all for the timetable to be valid (e.g. a room cannot be used for more than one purpose at any given time). Soft constraints are ones that should be satisfied if possible, but it is not completely necessary (e.g. grouping students lectures into blocks so that they don’t have lectures with large time gaps in between). I aim for the program to generate timetables that satisfy every hard constraint, and as many soft constraints as possible without causing an impractical run-time.

2. Project Aims and Objectives

My main aim is to develop a program which takes a list of constraints as input, and outputs a human readable timetable in a practical time limit (under two days). Below are some more detailed aims and objectives, split into primary and secondary aims and objectives. Primary aims and objectives are the ones that I definitely want to achieve. Secondary aims and objectives are the ones that I might achieve depending on available time, and depending on the direction of focus that my project takes.

2.1. Primary aims and objectives

- The program must output a timetable which violates zero hard constraints, and as little soft constraints as possible.
- Take as input a config text file containing hard and soft constraints.
- Output the timetable in human readable manner.
- Timetable should be generated in a practical time frame (less than 2 days)

2.2. Secondary aims and objectives (time permitting)

- Add functionality for parallel processing on different machines (splitting the timetable generation workload between different machines) via SSH.
- Create a separate GUI program that can be used to generate the constraint configuration file.
- Allow for the soft constraints to be weighted in terms of importance.

2.3. Constraints

I use the word "commitments" below to refer to: Lectures, labs, tutorials etc.

Hard

- Students must not have more than one commitment at a time.
- Lecturers must not instruct more than one commitment at a time.
- The room allocated must be large enough to accommodate all students allocated to it.
- A room must only be allocated to one purpose at once.
- Students and Lectures must have a minimum break of 1 period a day.
- Lectures must only have commitments assigned on days that they work (if they don't work Monday to Friday).
- Commitments must only be scheduled between Monday and Friday.

Soft

- Student commitments are in blocks, not spread out (convenience for students).
- Free days for students and lectures should be created where possible (by grouping commitments into the other days).
- No 9am commitments.
- Commitments should be finished for the day as early as possible without starting at 9am.

2.4. Testing

- Use timing based testing to ensure timetables are produced in under the two-day time limit.
- Use timing based testing to measure improvements made by implementation of new techniques/features.
- Also use constraint satisfaction in the generated timetable as a measure of the effectiveness of new techniques/features (the more constraints satisfied the better more effective the current state of the program).
- If enough time, test the current algorithm used against others to test which is the fastest.

3. Work Plan

3.1. Milestones

Each of the milestones below relates to milestones mentioned in the weekly aims and objectives planning section.

1. **Week Number 5** - Develop the program to a point where a timetable is produced for ten students, two lectures and two modules. All hard constraints must be satisfied. This milestone will ensure that I have the basic skeleton algorithm working, allowing me to begin on improvements.
2. **Week Number 9** - Develop the program to a point where a timetable is produced which can produce a timetable for one hundred students, five lecturers and five modules. This milestone will ensure that the program can produce a timetable good enough to be used in a practical scenario. From here I can decide to keep improving the algorithm, or add additional aforementioned features such as parallel processing.
3. **Week Number 12** - Finish development on the program as well as testing. This will leave me enough time to finish writing up the final report.

3.2. Weekly Aims and Objectives

Week Number	Week Starting	Aims and Milestones
1	30/01/2017	● Do background research on what type of algorithm to use, and potential methods to optimise the selected algorithm ● Research and arrive at a decision on which coding language I will use to develop the program.
2	06/02/2017	● Represent the problem and clearly determine how I intend to progress towards solving it.
3	13/02/2017	● Design the syntax for the configuration file. ● Code functionality to load configuration file into the program. ● Begin development of the solving algorithm.
4	20/02/2017	● Continue development of the solving algorithm.
5	27/02/2017	● Code functionality to export timetable in a file which is human readable. ● Continue development of the solving algorithm. ● **MILESTONE 1** ● Review meeting with Frank.
6	06/03/2017	● Begin improvements to solving algorithm. ● Begin testing in order to gauge improvement caused by new features.
7	13/03/2017	● Continue improvements to solving algorithm. ● Continue testing/comparing test results with previously obtained to measure improvement.
8	20/03/2017	● Continue improvements to solving algorithm. ● Continue testing/comparing test results with previously obtained to measure improvement.
9	27/03/2017	● Continue improvements to solving algorithm. ● Continue testing/comparing test results with previously obtained to measure improvement. ● **MILESTONE 2** ● Review meeting with Frank.
10	03/04/2017	● Continue development on the program. ● Continue testing.

11	10/04/2017	● Continue development on the program. ● Continue testing.
12	17/04/2017	● Finish development of the program ● **MILESTONE 3** ● Review meeting with Frank.
13	24/04/2017	● Work on writing up the final report
14	01/05/2017	● Finish writing up the final report ● **SUBMIT FINAL REPORT ON 05/05/2017**

3.3. Work Plan Extras

- Every week I will be meeting with Frank on Friday at 11:00am. On the weeks where I have planned a review meeting, it will take place in this time slot.
- I have allowed myself more development time than I expect I will need incase I encounter bugs that take a long time to resolve.

3.4. Deliverables

1. Final report as a pdf
2. Runnable program
3. Any files used for final testing in order to help showcase the program
4. Any code that may have been developed for additional features (e.g. code for SSH parallel processing)

3.5. Background Research Goals

- Research existing problems experienced with previous attempts at solving the timetabling problem.
- Research the different existing approaches to the timetabling problem, and each of their advantages and disadvantages.
- Compare different approaches and justify why I will be using an evolutionary algorithm as my approach.
- Research and decide on the type of evolutionary algorithm I will use.
- Research existing methods for optimising timetabling problems and identify methods applicable to me that I can take advantage of.
- Research which programming language would be suited best for the development of the program.

3.6. Ethical issues

Me and Frank foresee no ethical issues created by my project. There is no form of data protection that I need to consider and no possible way in which it could cause offense.