CS255 Artificial Intelligence Coursework - Timetabling

October 1, 2019

In this assignment, your goal is to produce a timetable for a week of lectures, given a set of tutors and several other constraints. You will produce three timetables, one focused only on module slots, another that considers lab sessions and a third that considers cost. This document will discuss the requirements of the timetable, and how to submit your solution. Your solution will be written in Python 3. **Please ensure that your solution runs on the DCS machines**. The deadline for this assignment is the **14th of January 2020**.

1 Getting Started

To begin, download the zip file from the CS255 web page, it will be underneath the heading 'Coursework'. Inside this zip file there are a number python files and a folder of text files, labelled ExampleProblems. These python files provide everything needed to load a set of resources, and develop your own strategies for creating a valid timetable. The ExampleProblems folder contains several text file representation of tutors and modules that can be used to test your solutions. There is also a readme.txt, which will explain the purpose of each file in sufficient detail. Please read through that file carefully. The two main python files that you will use in this assignment, runScheduler.py and scheduler.py both have preambles and comments, discussing how to use them. We have also provided a method that creates a random schedule, but does not check if it is legal. This should help demonstrate how to use the main methods and objects needed for this task.

1.1 runScheduler.py

This file is provided to allow you to test your strategies. Several pre-designed problems are included, allowing you to test your solutions in several environments. To run this file, you will need to use the command "`python3 runScheduler.py`". This will load the Scheduler class found in scheduler.py and use the createSchedule method in that class to provide a legal assignment of tutors to modules and modules to slots. It will output a Timetable object, which will then be checked to see that it satisfies every constraint. In addition to changing the problem number that is loaded when tested, you may also wish to change the scheduler method called from createSchedule to createLabSchedule or createMinCostSchedule, to allow you to test your solution for all three tasks.

1.2 scheduler.py

This file contains the Scheduler class, and the createSchedule, createLabSchedule and creatMinCostSchedule methods. These methods are where you will create your schedulers to fulfil the tasks described below. The preamble provided discusses the methods from other classes you are allowed to use in your solution. Importantly, you may only use the modules already imported into the file, which includes the python math and random libraries. Use of any other libraries, or methods from the provided files not explicitly stated in the preamble is not permitted and will result in a penalty.

Your createSchedule, createLabSchedule and createMinCostSchedule methods should assign a tutor and a module to each slot available. This timetable object should then be returned at the end of your methods, so that the timetable can then be evaluated to ensure that it follows the requirements discussed below.

2 Timetable Scheduling Programming Tasks (70% of assignment total)

You will be given a list of tutors and a list of modules. Each tutor consists of a name, and a list of expertise topics. Each module consists of a name, and a list of topic areas covered in that module. The timetable covers the 5 weekdays, Monday, Tuesday, Wednesday, Thursday and Friday. For task 1, each day has 5 slots, numbered 1 to 5. For task 2 and 3, this is extended to 10 slots, numbered 1 to 10.

You will be making three difference schedulers, as described below.

2.1 Task 1: Basic Scheduling (20% of assignment)

In this task, you must complete the createSchedule method in scheduler.py, to produce a schedule that adheres to the following requirements:

- Each module must be assigned to a slot.
- Each slot requires a tutor to be assigned to it. Assigning a tutor and a module to the same slot means that the tutor will be teaching that module.
- A tutor can only teach a module if the tutor's expertise topics include every topic covered by the module.
- A tutor can only teach a maximum of two different modules.
- A tutor cannot teach multiple modules in a single day.

Hint: Consider the techniques discussed in the lectures, such as backtracking and how they may help with this task.

2.2 Task 2: Introducing Lab Sessions (20% of assignment)

The school is modifying its curriculum, and introducing lab sessions. Each day now has 10 slots, and lab sessions require a lab tutor. You must now create a scheduler that adheres to the following requirements:

- Each module must be assigned to a slot.
- Each lab session must be assigned to a slot.
- Each slot, module or lab session, requires a tutor to be assigned to it.
- A tutor can only teach a module if the tutor's expertise topics include every topic covered by the module.
- A tutor can teach a lab session if their expertise topics include at least one topic covered by the module.
- A tutor can only teach a maximum of 4 credits. Modules count as 2 credits, lab sessions count as 1 credit.
- A tutor cannot teach more than 2 credits in a day.

You must complete the createLabSchedule method in scheduler.py with your solution to this problem.

2.3 Task 3: Cost-effective Scheduling (30% of assignment)

In this task, you must now consider the cost of a schedule. To hire a tutor to teach a single module costs £500. If that tutor is hired for a second module, the second module will only cost £300. If the two modules are on consecutive days, the second module only costs £100. As such, it is preferable to hire a tutor to teach two modules, and place them on consecutive days.

To hire a tutor to teach a single lab session costs £250. Each subsequent lab session a tutor teaches costs £50 less, so the second session will cost £200, the third £150 and the fourth £100. Furthermore, a lab session that is taught on the same day as something else the tutor is teaching, lab session or module, has its cost halved. This means if a tutor is teaching two labs on the same day, they would cost £125 and £100 respectively instead of £250 and £200.

You must complete the createMinCostSchedule method in scheduler.py, to produce a schedule of the minimal possible cost.

Hint: You should consider a heuristic based approach, think of what you can use to measure closeness to the optimal solution.

3 Report (30% of assignment)

In addition to your scheduler class, you must also submit a report. This report should be written in the IEEE format (a LaTeX template is provided on the CS255 web page), and should not exceed 3 sides. It should detail your approach for each part of the programming task, including justification and evaluation, which should include some discussion about how your approaches compare to the optimal solution. Your report will be marked on technical content, evaluation and the quality of writing.

4 Submission

You will submit two files to Tabula. The first file should contain the Scheduler class, with the createSchedule, createLabSchedule and createMinCostSchedule methods defined. Please rename the file to uniID.py, e.g. 1003685.py. The second file should be your report as a PDF generated from the LaTeX template.

Ensure that your submitted file can solve each task, does not use any methods that are not permitted in the preamble and contains no additional imports.

Furthermore, please ensure that your solution runs in a *reasonable time frame*. A solution should not run for longer than 5-10 minutes.