Report on Timetabling

CS255 Artificial Intelligence Coursework

University ID: 1809282

Date: 13-01-2020

I. Introduction

This report describes my approach and analysis of each task of the coursework on a constraint satisfaction problem.

II. Task 1 and 2

I will discuss the methods that are scheduling Task 1 and Task 2 under the same section as the main difference between them is the implementation - The eligibility of a tutor to teach a module and a lab is checked in a different way, which has to be considered in Task 2. In every occasion when an action with modules and labs is described, in case of Task 1 the action is only executed on the modules as labs are not necessary to be timetabled.

A. Design choices

First, a matching of tutors to modules/labs is created, where every module is assigned a tutor such that every match satisfies 2 constraints: each tutor is eligible to teach each module and lab they have been assigned to and no tutor teaches more credits than the allowed 4 credits (where a lab worth 1 credit and a module is worth 2 credits).

Then every match is assigned to a time slot such that no tutor teaches more than the allowed 2 credits per day.

The task of matching modules and tutors can be described as a constraint satisfaction problem where the modules/labs are the variables and tutors are the values assigned to them. Then an eligible matching is found with backtracking search. Although the domain of each variable is finite the search space is large. The efficiency of the backtracking algorithm can be greatly improved by using appropriate heuristics [1]:

Minimum Remaining Values – when choosing the next variable that should be assigned next the module/lab with the fewest legal values is chosen, that is the module/tutor which has the smallest number of tutors who are eligible to teach them. This is achieved by ordering the modules/labs in a shared list by the number of tutors that are eligible to teach them in increasing order and then always choosing the next first module/lab from the list that has not been assigned a tutor yet.

Least Constraining Value - the tutors (values) for a given variable are tried in an order such that the least constraining value is tried first, that is the tutor that rules out the fewest values tutors in the remaining modules. The tutor that is the least constraining is going to be the tutor whose expertise is includes the most topic. Choosing the correct tutor can be achieved by ordering the tutors in a list by the number of topics included in their expertise in decreasing order and then always choosing the first available tutor from the list.

These heuristics will ensure that the program runs in a reasonable time frame.

B. Implementation

A vaild timetable is created with the functions *modSchedule* in Task 1 and with *labMosSchedule* in Task 2. First they order the modules/labs and the tutors so that they satisfy the orders described in the heuristics above. Then they call the recursive functions *recurModMatching* and *recurLabModMatching* respectively, which both return an appropriate matching of modules and tutors. These recursive functions terminate when all modules/labs have been assigned a tutor.

These matches are then put into the timetable in order of the tutors.Starting with the modules/labs of one tutor the timetable is filled up row by row, first every day is filled with a match in the first row, then in the second row and so on, starting from Monday in every row. This will ensure that no tutor is teaching more that one module/lab on one day. As a tutor can teach at most 4 of labs/modules and if they are teaching these on consecutive days (with Friday and Monday being 'consecutive') they will not teach more than 2 credits on any given day.

III. Task 3

Task 3 is a modified version of Task 2, therefore I will not describe the parts of Task 3 that correspond to Task 2.

A. Design Choices

The idea of Task 3 is similar to that of Task 2: First a good initial matching of tutors to modules is produced and then these assignments are put into the timetable.

In addition to this for Task 3 a new heuristic is choosing tutors to improve an the cost of the schedule and the assignments are put into the timetable in a different way in order to achieve reductions of the cost.

There are many different heuristics that can be used to achieve a schedule with optimized cost. I implemented one version, but I will describe more ideas. One approach is to aim to have tutors who only teach modules and tutors who only teach labs. Sometimes it is not possible to distribute the modules like this, but the more tutors are there who teach only one type of session the smaller the cost is. This is because there is no cost reduction that needs teaching modules and labs as well, but there are reductions if multiple sessions of the same type are taught. The average cost of modules is the lowest if the maximum number (24) of modules are taught by tutors who are teaching another module. Then the average cost of a module is $24 * (500 + 100) + 500 = 596$. Similarly, the average cost of labs is the lowest if the maximum number (24) of the labs are taught by tutors who are teaching 3 other labs. Then the average cost of a lab is $6 * (125 + 100 + 75 + 50) + 250 = 94$ Figure 1 shows the minimum costs of a tutor who is teaching a certain number of modules and labs.

$Modules$	$Labs$	$MinCost$	$MaxCost$
2	0	600	800
1	0	500	500
1	1	750	750
1	2	725	950
0	1	250	250
0	2	225	450
0	3	325	600
0	4	350	700

Fig. 1. Min and max cost of hiring a tutor for given number of modules/labs

Another approach would be to aim to maximize the credits each tutor is teaching, this again brings the minimum cost closer as hiring a new tutor is always at least as expensive as giving another module to an old tutor.

B. Implementation

The first described heuristic is implemented in the following way: when choosing the tutors to try out for a given module first try out tutors who are already teaching modules, then tutors who are not teaching anything then tutors who are teaching labs. (Tutors who are teaching both a module and labs will not be eligible to teach any more modules) When choosing the tutors to try out for a given lab first try out tutors who are already teaching labs, then tutors who are teaching both labs and a module, then tutors who are not teaching anything and then tutors who are teaching a lab.
The optimal way in which the assignments can be put into a timetable is achieved if every two module taught by the same tutor is on consecutive days, every 4 labs that are taught by the same tutor are distributed between 2 days, in case of a tutor teaching 3 labs 2 of those are on the same day and the 3rd lab is on a day which is after that and in case of a tutor teaching 2 labs those are on the same day.

The implemented timetabling does not optimize the case of 3 labs, the 3rd lab may be on an earlier day than the 2 labs on the same day, however, even like this the average cost (of the 8 given example) is reduced by 18%

Put the assignments into the timetable in the following way: Let's reserve the slots denoted by M to modules and the slots denoted by L for labs in Figure 2. The index represents the ordering of the slots, when a module/lab is put into the next slot it means the next available slot in order. The modules will be put in horizontally paired, the modules of the same tutor will be on consecutive days. Similarly the labs will be put in vertically paired, those labs will have reduced cost. (The vertical pairs will follow each other horizontally in the ordering. The slot L_{25} will be the last one, but just for convenience, the order of the slots within a day does not matter. The labs the are in pairs will be put in from forwards in pairs, and the labs that are standing alone will be put in from backwards. (Except at few cases if the lab's tutor is also teaching a module) Then the assignments in order:

	M	T	W	T	F
1	M_1	M_2	M_{21}	M_{22}	M_{25}
2	M_3	M_4	M_{23}	M_{24}	L_{25}
3	M_5	M_6	L_1	L_3	L_5
4	M_7	M_8	L_2	L_4	L_6
5	M_9	M_{10}	L_7	L_9	L_{11}
6	M_{11}	M_{12}	L_8	L_{10}	L_{12}
7	M_{13}	M_{14}	L_{13}	L_{15}	L_{17}
8	M_{15}	M_{16}	L_{14}	L_{16}	L_{18}
9	M_{17}	M_{18}	L_{19}	L_{21}	L_{23}
10	M_{19}	M_{20}	L_{20}	L_{22}	L_{24}

Fig. 2. Slots reserved to modules (**M**) and to labs (**L**)

- Put in the modules of the tutors who are teaching 2 modules
- Put in the modules of the tutors who are teaching 1 module and if they are teaching any labs put them in in the following way: if they teach 2 labs find the first double slot from backwards which is on a day that does not collide with the module's day and put in the labs, if they teach 1 lab put in to the first slot going from backwards that is not on the same day as the module is($L_{25}, L_{24}...$)
- Put in the labs of the tutors who are teaching 4 labs into the next two double slots
- Put in the labs of the tutors who are teaching 3 labs. First put in 2 labs in the next double slot, then from backwards to the first slot which is not on the day of the double slot.
- Put in the labs of the tutors who are teaching exactly 2 labs (and no modules!) to the next available double slot
- Put in the labs that are taught by a tutor who only teaches 1 lab (and no modules!)

C. Analysis

The new way of putting assignments into the timetable has improved the average cost of the 8 problems by 18% as Figure 3 shows. The heuristic I implemented on top of this improves the cost as Figure 3 shows, it gets close to the optimal solutions costing 10050. However, it comes with a large decrease in efficiency. All of the 8 example problems terminate under 5 minutes (all together). Problem 1 and especially Problem 4 are running for quite a long time compared to how long it takes

$Prob.$	$Task2$	$Task3$	$Imp.$	$Ratio$	$Heur$
1	13550	11550	2000	0.15	10375
2	15100	12375	2725	0.18	11175
3	14350	11950	2400	0.17	10450
4	14150	11150	3000	0.21	10950
5	14350	11550	2800	0.20	10450
6	14350	11850	2500	0.17	11250
7	14750	11950	2800	0.19	12025
8	15800	13025	2775	0.18	10950
$Av.$	14550	11925	2625	0.18	10953.125

Fig. 3. Improvement on costs in Task 3

without the heuristics. Since the improvement of the cost is not too large in a real life situation I believe that my solution without the extra heuristic is more useful than this. That is why I commented out the heuristic in scheduler.py, I indicated which parts of the code needs to be commented in and out in order to have the slower solution, which provides a lower cost. Important to note that on average this heuristic provides improvement on cost, but some problems (Problem 7) will have an increased cost. The average cost has been improved by almost 1000 with the heuristic, which is not as significant as the average improvement of 2625 when introducing the new way of timetabling.
Comment in: lines 518-556, 560
Comment out: line 563

References

[1] N. Griffiths, 2019, CS255 Artificial Intelligence: Constraint Satisfaction Problem, lecture notes, pp.12-16, University of Warwick