Simulation Tools and Techniques

University Course Timetabling Problem

May 15, 2017

Authors: Ján Profant, xprofa00@stud.fit.vutbr.cz

Faculty of Information Technology
Brno University of Technology

Contents

Assignment

In this paper, we present hybrid approach to university course timetabling problem based on [3]. Timetabling problem proposed here has been designed by Ben Paechter and it is basically the reduction of typical university timetabling problem. We used hybrid methodology to create initial population, then we used mutation process and randomized iterative improvement to create solution with the lowest penalty as possible.

Problem Definition

The timetabling problem proposed here has been designed by Ben Paechter for the Metaheuristics Network. It is a reduction of a typical university course timetabling problem. It consists of a set of events to be scheduled in 45 timeslots (5 days of 9 hours each), a set of rooms in which events can take place, a set of students who attend the events, and a set of features satisfied by rooms and required by events. Each student attends a number of events and each room has a size [3].

A feasible timetable is one in which all events have been assigned a timeslot and a room so that the following hard constraints are satisfied:

- no student attends more than one event at the same time
- the room is big enough for all the attending students and satisfies all the features required by the event
- only one event is in each room at any timeslot

In addition, a candidate timetable is penalized equally for each occurrence of the following soft constraint violations:

- count the number of occurences of a student having just one class on a day (e.g. count 2 if a student has two days with only one class)
- count the number of occurrences of a student having more than two classes consecutively (3 consecutively scores 1, 4 consecutively scores 2, 5 consecutively scores 3, etc), classes at the end of the day followed by classes at the beginning of the next day do not count as consecutive
- count the number of occurrences of a student having a class in the last timeslot of the day

Sum the three counts to give the solution score - smaller is better - zero is always possible with the instances in this competition [9].

Initial Population

For creating an initial population, we used hybrid methodology [8], because we were not able to implement solution which could give us initial solution based on [3] - a construction algorithm is used in [3], to generate large populations of random feasible timetables. The approach, which starts with an empty timetable, is similar to a random graph colouring method. A feasible solution is obtained by adding or removing appropriate courses from the schedule until the hard constraints are met. A roulette wheel selection is employed to select individuals for the new population. As already mentioned, we were not able to implement this solution, the size of population is in the article set to 100 and the time complexity of this solution would be too high. Authors noted important thing - the experiments were run for 200000 iterations which take approximately ten hours for each of the datasets. Note that course timetabling is a problem that is usually tackled several months before the schedule is required. A ten hour run for course timetabling is perfectly acceptable in a real world environment. This is a scheduling problem where the time taken to solve the problem is not critical [3].

Largest Degree

Graph heuristics are widely studied methods which were developed during the early days of research on timetabling problems. They are used in sequential (or constructive) solution methods to order the events that are not yet scheduled according to the difficulties of scheduling them into a feasible timeslot (without violating any hard constraints). The difficulties are represented by the degrees of the vertices in the graph, which model the timetabling problem by representing the events as vertices and conflicts by edges. We say two events in timetabling problems have a conflict if they involve the same students. The difficulty of an event is represented by the number of conflicts it has with the others. The objective is to construct a timetable by scheduling the most conflicting events one by one into feasible timeslots, satisfying as many of the soft constraints as possible [7].

Once all events have been assigned to timeslots, the maximum matching algorithm for bipartite graph is used to assign events to room considering all the features. After this step, there is no guarantee, that solution given is feasible [8].

Local Search

We employ two neighbourhood movements in this step. Move one (**M1**) selects one event at random and assigns it to a feasible pair timeslot-room also chosen at random. Move two (**M2**) selects two events at random and swaps their timeslots and rooms while ensuring feasibility is maintained. A move is only accepted if it improves the satisfaction of hard constraints - the number of feasible courses. This step terminates if no move produces a better (closer to feasibility) solution for 10 iterations [8].

Mutation

We carry out a random light mutation on 20% of the courses from 20% of the selected individuals. Courses are chosen at random from any point and are reallocated to the next earliest feasible timeslots [3].

Randomized Iterative Improvement

This algorithm always accepts an improved solution and a worse solution is accepted with a certain probability. It uses eleven neighborhood structures [4]:

1. Select two courses at random and swap timeslots.

2. Choose a single course at random and move to a new random feasible timeslot.

3. Select two timeslots at random and simply swap all the courses in one timeslot with all the courses in the other timeslot.

4. Take 2 timeslots (selected at random), say t_i and t_j (where $j > i$) where the timeslots are ordered $t_1, t_2, ..., t_{45}$. Take all the exams in t_i and allocate them to t_j. Now take the exams that were in t_j and allocate them to t_{j-1}. Then allocate those that were in t_{j-1} to t_{j-2} and so on until we allocate those that were in t_{i+1} to t_i and terminate the process.

5. Move the highest penalty course from a random 10% selection of the courses to a random feasible timeslot.

6. Carry out the same process as in fifth but with 20% of the courses.

7. Move the highest penalty course from a random 10% selection of the courses to a new feasible timeslot which can generate the lowest penalty cost.

8. Carry out the same process as in N7 but with 20% of the courses.

9. Select one course at random, select a timeslot at random (distinct from the one that was assigned to the selected course) and then apply the kempe chain [11]. This approach was not successfully implemented.

10. This is the same as N9 except the highest penalty course from 5% selection of the courses is selected at random.

11. Carry out the same process as in N9 but with 20% of the courses.

Experiments

We implemented our solution in python2. As already mentioned, we were not able to implement approach for creating initial population in [3], so we used graph algorithms and local search operator presented in [8]. We used datasets from timetabling competition [1] and datasets generated by Paechter's generator [1]. In our experiments, we used only small datasets, because our solution implemented in python2 is much slower than reference from [3] using Visual C++ and we used only 20000 iterations of iterative randomized improvement comparing to 200000 in [3]. Table 1 presents our best achieved results on benchmark datasets comparing to different approaches.

Dataset	M1	M2	M3	M4	M5	M6	M7	M8	M9
*small*1	**4**	0	0	0	8	1	1	6	10
*small*2	**7**	0	0	0	11	3	2	7	9
*small*3	**4**	0	0	0	8	1	0	3	7
*small*4	**7**	0	0	0	7	1	1	3	17
*small*5	**1**	0	0	0	5	0	0	4	7

Table 1: Legend: M1 (our approach - best), M2 (best) [3], M3 (best) [4], M4 (best) [2], M5 (local search - average) [10], M6 (ant algorithm - average) [10], M7 (best) [6], M8 (best) [7], M9 (best) [5]

[1]http://www.soc.napier.ac.uk/ benp/

Discussion and Conclusions

We can see, that our approach is better than M5 [10], M9 [5] and can be compared to solutions produced by M8 [7] on small datasets. Results are little bit different comparing to our reference paper [3], especially because we did not use Kempe chain [11] and we used only 10% of iterations. We expect, that our solutions will work also on larger datasets, but our implementation is very slow due to programming language choice and many other implementation details. We managed to use another approach for creating initial population, we used local search in creating initial population, then we applied mutation algorithm and iterative randomized improvement algorithm to find the most valuable solution as possible.

Bibliography

[1] International timetabling competition. http://sferics.idsia.ch/Files/ttcomp2002, 2002.

[2] Salwani Abdullah, Edmund K Burke, and Barry Mccollum. An investigation of variable neighbourhood search for university course timetabling. In *The 2nd multidisciplinary international conference on scheduling: theory and applications (MISTA)*, pages 413–427, 2005.

[3] Salwani Abdullah, Edmund K Burke, and Barry McCollum. A hybrid evolutionary approach to the university course timetabling problem. In *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pages 1764–1768. IEEE, 2007.

[4] Salwani Abdullah, Edmund K Burke, and Barry McCollum. Using a randomised iterative improvement algorithm with composite neighbourhood structures for the university course timetabling problem. In *Metaheuristics*, pages 153–169. Springer, 2007.

[5] Hishammuddin Asmuni, Edmund K Burke, and Jonathan M Garibaldi. Fuzzy multiple heuristic ordering for course timetabling. In *Proceedings of the 5th United Kingdom workshop on computational intelligence (UKCI 2005)*, pages 302–309. Citeseer, 2005.

[6] Edmund K Burke, Graham Kendall, and Eric Soubeiga. A tabu-search hyperheuristic for timetabling and rostering. *Journal of heuristics*, 9(6):451–470, 2003.

[7] Edmund K Burke, Barry McCollum, Amnon Meisels, Sanja Petrovic, and Rong Qu. A graph-based hyper-heuristic for educational timetabling problems. *European Journal of Operational Research*, 176(1):177–192, 2007.

[8] Dario Landa-Silva and Joe Henry Obit. Comparing hybrid constructive heuristics for university course timetabling. 2011.

[9] Olivia Rossi-Doria, Michael Sampels, Mauro Birattari, Marco Chiarandini, Marco Dorigo, Luca M Gambardella, Joshua Knowles, Max Manfrin, Monaldo Mastrolilli, Ben Paechter, et al. A comparison of the performance of different metaheuristics on the timetabling problem. In *International Conference on the Practice and Theory of Automated Timetabling*, pages 329–351. Springer, 2002.

[10] Krzysztof Socha, Joshua Knowles, and Michael Sampels. A max-min ant system for the university course timetabling problem. In *International Workshop on Ant Algorithms*, pages 1–13. Springer, 2002.

[11] Jonathan M Thompson and Kathryn A Dowsland. Variants of simulated annealing for the examination timetabling problem. *Annals of Operations research*, 63(1):105–128, 1996.