Tackling the university course timetabling problem with an aggregation approach

Mieke Adriaen, Patrick De Causmaecker†, Peter Demeester
and Greet Vanden Berghe

KaHo Sint-Lieven
Information Technology
Gebroeders Desmetstraat 1, 9000 Gent, Belgium
{Mieke.Adriaen, Peter.Demeester, Greet.VandenBerghe}@kahosl.be

† Katholieke Universiteit Leuven Campus Kortrijk
Computer Science and Information Technology
Etienne Sabbelaan 53, 8500 Kortrijk, Belgium
Patrick.DeCausmaecker@kuleuven-kortrijk.be

1 Introduction

In this abstract we describe the university timetabling problem as it is perceived and solved at the KaHo Sint-Lieven School of Engineering. Timetabling is carried out manually by dragging and dropping events in a room-timeslot matrix, but potential conflicts are automatically spotted by the conflict module built into the application. We are now in the process of automatizing the construction of timetables itself. In the next sections we describe the specific timetabling problem at KaHo Sint-Lieven and the steps we follow to tackle it. The approach is based on a tabu search algorithm. Inspired by Kingston [13], we group sessions in order to make the timetabling problem less complex.

2 University Course Timetabling

University timetabling is probably one of the best studied timetabling problems [2,4,6,14] in academia. Bardadym [2] classifies university timetabling into 5 groups.

- faculty timetabling: assign qualified teachers to courses,
- class-teacher timetabling: assign courses with the smallest timetabling unit being a class of students,
- course scheduling: assign courses with the smallest scheduling unit being an individual student,
- examination scheduling: assign examinations to students such that students do not have two examinations at the same moment,
- classroom assignment: after assigning classes to teachers, assign these class-teacher couples to classrooms.

E. K. Burke, H. Rudová (Eds.): PATAT 2006, pp. 330–335. ISBN 80-210-3726-1.

The problem considered in this abstract can be classified as course scheduling. That problem consists of placing events (which we will call sessions) in appropriate class rooms and timeslots, taking into account that the number of students attending a session has to fit into the room and that the duration of the session cannot be greater than the length of the timeslot it is assigned to. Other constraints that have to be taken into consideration are that students and lecturers cannot be at two places in the same timeslot and that a session can only be assigned to one timeslot and one room. The university timetabling problem that is studied in this paper is in general more complex than the secondary school timetabling problem. In the latter, timeslots are equal in length and the schedule is weekly repeated during a semester. At the KaHo Sint-Lieven School of Engineering, a timetable is typically constructed for a period of a semester (which is twelve weeks long). The problem however is that some subjects are taught every week; some sessions are taught every fortnight, others only the first six weeks of the semester, others only nine times a semester, ... It is even more complex since the timeslots are not equal in length, and since they can overlap. This has to be taken into consideration when constructing the timetable. Multiple lecturers can be assigned to lab sessions which are taken by large student groups. This makes the manual construction of timetables a hard and time consuming task. It usually takes a couple of days to remove all the conflicts in the timetable that appear once the semester has started.
The considered problem size for one semester is:

- 12 weeks,
- 133 different teacher groups,
- 212 different student groups,
- there are 100 class rooms of different sizes available,
- there are 20 different (sometimes overlapping) possible timeslots per day,
- 1215 sessions need to be scheduled.

Most university timetabling applications described in the literature have been developed by universities that experienced the need for automatizing their own timetabling problem [6]. Bardadym [2] remarks that most of these systems are usually very problem specific and can only be applied in that particular university. We are attempting to overcome that by developing a general timetabling framework [7].

3 Tabu Search Approach

The literature about university course timetabling teaches us that researchers applied different approaches to tackle the problem (see [14] for an overview). Meta-heuristics approaches are known to produce good results. Early papers (beginning 1990's) of Hertz [11,12] report good results applying tabu search [9] on the university course timetabling problem. More recent papers on university course timetabling combine tabu search with other search methods: Schaerf [15] applies a combination of randomized hill-climbing with tabu search, while Burke

et al. [5] use a hyperheurisitic in which a tabu search algorithm is employed to select the lower level heuristics.

3.1 General Description of the OpenTS Framework

In this contribution we report on the OpenTS[10] framework and its employability on university course timetabling. That framework contains a Java based implementation of a tabu search algorithm. Developers who start applying the framework are expected to implement the problem specific interfaces. They thus describe how a solution will be represented, what will be the different neighborhoods, and how the different constraints will be evaluated. The OpenTS framework also offers the opportunity to dynamically adapt the length of the tabu list (reactive tabu list) and to switch between different neighborhoods.

3.2 Application of OpenTS to the Course Timetabling Problem

In this section we describe how we applied the OpenTS framework to the university course timetabling problem. We implemented the problem specific interfaces and incorporated the possibility to dynamically adapt the tabu length size and switch between neighborhoods.

- The implementation of the OpenTS Solution interface consists in this case of a two dimensional matrix with the class rooms in the rows and the timeslots in the columns. If a session is planned, the matrix will contain a session ID, otherwise it will contain 0.
- We also implemented different moves which define neighborhoods. One move simply shifts a session ID to a matrix element that equals 0. If a selected element differs from 0, it swaps the two sessions. Another move swaps all the scheduled sessions in a particular timeslot with any other timeslot.
- The last interface that a user of OpenTS needs to implement is the Objective Function interface. By implementing this interface the developer decides how the different hard and soft constraints will be evaluated. To ascertain the quality of a solution proposed by the tabu search algorithm (how many constraints are violated by the proposed solution), this solution needs to be evaluated by the objective function. At this moment only the hard constraints described in the previous section are evaluated.
- If an iteration is unsuccessful - no better solution is found in the considered iteration - the tabu length is increased. If, however, a better solution is found the length of the tabu list is decreased (until the lower limit of size 7 is reached). We opt to choose primes as tabu lengths. Switching between neighborhoods happens when the number of unimproving moves in one neighborhood exceeds a predefined value.

The described method allows to schedule a weekly timetable. As noted in Section 2 however, some sessions are only taught a few times during a semester.

4 Aggregation

To keep the problem size small, we opt not to construct twelve independent weekly timetables, which probably would not differ much from week to week. Instead we decided to carry out a pre-processing step. The idea is to group lectures which are not organized on a weekly basis into aggregate sessions, which we call 'pillars'. If a lecture is only organized six times during a semester, the application will try to find a lecture that is also taught six times (or less). Fig. 1 shows a graphical representation of a semester consisting of twelve weeks. The 'pillar' in the middle of the figure is built up of two lectures organized in interleaved weeks (marked with and without a cross) that are each organized six times during a semester to the same class group. The evaluation method in this example only has to check the constraints for two consecutive weeks (assuming of course that these lectures are organized alternatingly).

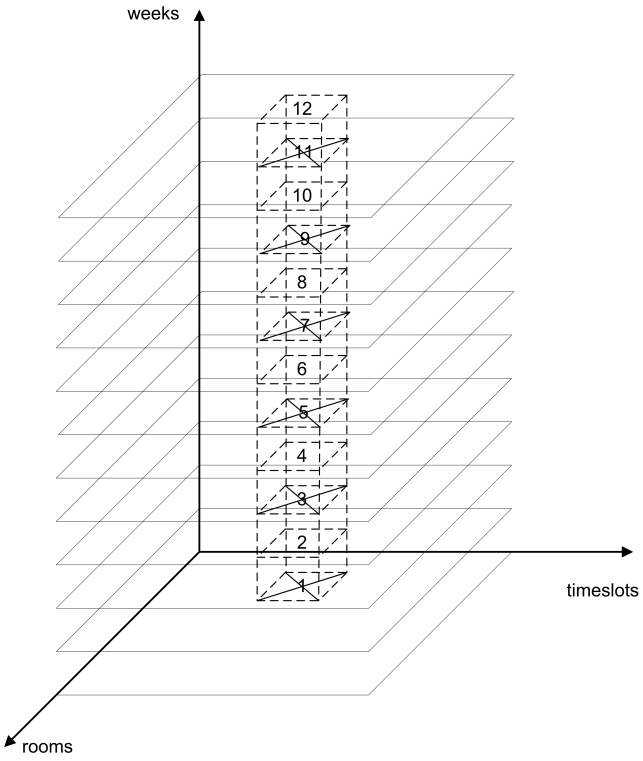

Fig. 1. Graphical representation of pillars. Sessions in the pillar are taught on a fortnightly basis.

The method is based upon the tiling approach that Kingston [13] successfully applies to Australian high school timetabling. He groups similar sessions and

treats them as one. We are also thinking about generalizing the pillars concept. In the case described above, the pillar can only be constructed for lectures organized to the same class group, using the same location and timeslot.

5 Results

The above described model is implemented and tested on real-world data. The preliminary results look rather promising.

6 Conclusion and Future Work

The presented model with the pillar representation offers a good quality method to solve large timetabling problems. Currently, it only deals with hard constraints. Remaining problems to solve are:

- investigating if the pillar representation can be generalized and applied to sessions that only have lecturers or students in common.
- taking into account soft constraints (e.g. avoid sessions on the last timeslot of the day, avoid free hours, take into account personal preferences of lecturers,...). One candidate approach is the linear numberings method [3,8] that was first introduced for employee timetabling. In that linear numberings method constraints are expressed in terms of eight numbering constraints and a corresponding template of numbers.

References

1. R. Alvarez-Valdes, E. Crespo, J.M. Tamarit: Design and implementation of a course scheduling system using Tabu Search, European Journal of Operational Research, Volume 137, Number 3, 512-523(12), 2002
2. V.A. Bardadym: Computer-Aided School and University Timetabling: The New Wave, Selected and Revised Papers of the 1st International Conference on Practice and Theory of Automated Timetabling, (PATAT 1995), Edinburgh, Springer LNCS 1153, 22-45, 1996
3. E.K. Burke., P. De Causmaecker, S. Petrovic, G. Vanden Berghe: Fitness Evaluation for Nurse Scheduling Problems, In Proceedings of Congress on Evolutionary Computation, CEC2001, Seoul, IEEE Press, 1139-1146, 2001
4. E.K. Burke, K.S. Jackson, J.H. Kingston and R.F. Weare: Automated Timetabling: The State of the Art, The Computer Journal, Vol. 40, No. 9, 565-571, 1997
5. E. Burke, G. Kendall, E. Soubeiga: A tabu-search hyperheuristic for timetabling and rostering, Journal of Heuristics, 9, 451-470, 2003
6. M.W. Carter and G. Laporte: Recent Development in Practical Course Timetabling, Selected and Revised Papers of the 2nd International Conference on Practice and Theory of Automated Timetabling, (PATAT 1997), Toronto, Springer LNCS 1408, 3-19, 1998

7. N. Custers, P. De Causmaecker, P. Demeester and G. Vanden Berghe: Semantic Components for Timetabling, Selected and Revised Papers of the 5th International Conference on Practice and Theory of Automated Timetabling, (PATAT 2004), Pittsburgh, Springer LNCS 3616, 17-33, 2005
8. P. De Causmaecker, P. Demeester, G. Vanden Berghe, B. Verbeke: Evaluation of Employee Rosters with the Extended Linear Numberings Method, Proceedings of UK PlanSIG, London, 125-133, 2005
9. F. Glover, M. Laguna: Tabu Search, Kluwer, Boston, 1997
10. R. Harder: OpenTS - Java Tabu Search http://www-124.ibm.com/developerworks/oss/coin/OpenTS/index.html
11. A. Hertz: Tabu Search for Large Scale Timetabling Problems, European Journal of Operational Research, Vol. 54, 39-47, 1991
12. A.Hertz: Finding a feasible course schedule using Tabu search, Discrete Applied Mathematics, Vol. 35, 255-270, 1992
13. J. H. Kingston: A Tiling Algorithm for High School Timetabling, Selected and Revised Papers of the 5th International Conference on Practice and Theory of Automated Timetabling, (PATAT 2004), Pittsburgh, Springer LNCS 3616, 208-225, 2005
14. S. Petrovic and E. Burke: University Timetabling, Handbook of Scheduling: Algorithms, Models, and Performance Analysis (ed. J. Leung), CRC Press, Chapter 45, 45-1 - 45-23, 2004.
15. A. Schaerf: Local search techniques for large high-school timetabling problems, IEEE Transactions on Systems, Man, and Cybernetics- Part A: Systems and Humans, 29(4), 368-377, 1999