The Second International Timetabling Competition (ITC-2007): Curriculum-based Course Timetabling Track

— preliminary presentation —

Luca Di Gaspero[1], Barry McCollum[2], and Andrea Schaerf[1]

[1] DIEGM, University of Udine
via delle Scienze 208, I-33100, Udine, Italy
{l.digaspero | schaerf}@uniud.it
[2] School of Electronics, Electrical Engineering and Computer Science,
Queen's University
SARC Building, Belfast, United Kingdom
b.mccollum@qub.ac.uk

Abstract. Following the success of the First International Timetabling Competition in 2002, the timetabling research community is organising a new competition on this problem (opening June 1st). This new competition will be on three different timetabling problems, and one of the tracks concerns the course timetabling formulation that applies to Italian universities (called Curriculum-based Course Timetabling). The dataset is composed by real-world instances provided by the University of Udine. In this work, we overview the general rules of the competition and we describe in details the problem formulation and the instances proposed for this track.

1 Introduction

Timetabling within Universities has long been recognised as a difficult combinatorial problem of practical relevance. Whether it be timetabling exams or courses, much (human or computing) effort is spent for producing solutions that are workable and of a high quality (see, e.g., [5]).

The First International Timetabling Competition (ITC-2002), which was organised by the International Metaheuristic Network, attracted 24 feasible submissions from all over the world. Information relating to the problem definition, instances, rules, and solution evaluation of ITC-2002 is available at the webpage http://www.idsia.ch/Files/ttcomp2002/.

The ITC-2002 was based on a problem formulation specifically proposed for the competition, and on a set of artificially generated instances, but it contained many of the characteristics found in certain university settings. Thanks to the competition, this formulation has successively become a standard, and many researchers have used it within their work (see, e.g., [1, 3, 2]). ITC-2002 therefore

had a positive effect of generating common ground for cross-fertilisation of ideas within research groups in the timetabling community.

The Second International Timetabling Competition (ITC-2007), opening June 1st, follows the main ethos of the first edition, but also aims at advancing upon it in a number of respects. The first innovation consists in being subdivided in three tracks so as to better cover the main formulations of the field of educational timetabling problems. Specifically, the tracks will be on: *Examination timetabling*, *Post Enrolment Course Timetabling* (the evolution of the formulation of ITC-2002), and *Curriculum-based Course Timetabling*. The second innovation aims at bridging the gap between research and practice: the competition introduces a significant degree of complexity in all tracks so that the new formulations employed are closer (in more aspects, although not all) to those of 'real world' problems [4].

In this paper, we describe the Curriculum-based Course Timetabling track, which is under the responsibility of the authors. We present the general rules of ITC-2007, the problem formulation of the track, and the description of the instances. Some discussion closes the paper.

The information presented here are preliminary, because the organisation is still on-going and some rules still can change up to the actual opening of the competition. All updated information will appear in the ITC-2007 web site: `http://www.cs.qub.ac.uk/eventmap/`.

2 Competition rules

The competition will have a set of rules that the participants have to satisfy. The rules of ITC-2007 are in large part taken from those of ITC-2002, but obviously some modifications have been made based on the previous experience.

1. The competition has an opening day and a deadline (approximately 6 months later). All coding and experimenting must be finished by the deadline.
2. Participants have to implement an algorithm to tackle the problem on a single processor machine. It can be expressed using any programming language.
3. The goal is to produce feasible timetables, in which a number of hard constraints are satisfied, and to minimise the number of broken soft constraints. If feasibility cannot be reached, information outlined on the solution produced should be provided, and the number of violated hard constraints is used for evaluation.
4. The dataset will be split into three sets of instances:
   Early instances: A first set of instances will appear on the web at the opening of the competition.
   Late instances: A second set of instances will be published two weeks before the deadline.
   Hidden instances: A third set of instances will be revealed only after the competition has closed and the participants have submitted their solvers. These will be used internally to rank the solvers submitted.

5. All solvers will be granted a fixed CPU time. Participants have to benchmark their machine with a program provided to them in order to know for how much time they can run their program on their machines.
6. The algorithms should run on a single processor machine, take as input a problem file in the format described, and produce as output a feasible timetable (if found) with a minimum number of soft constraint violations in the allowed CPU time. The output timetable must adhere to the data format determined by the organisers.
   The algorithm should not take account of additional knowledge about the instance (e.g., results from previous runs). The same version (and fixed parameters) of the algorithm must be used for all instances. That is, the algorithm should not "know" which instance it is solving: although it might analyse the problem instance and set parameters accordingly, it should not "recognise" the particular instance.
7. The participants must be prepared to show that those results are repeatable in the given computer time. In particular the participants should make their program in such a way that the *exact* run that produced each solution submitted can be repeated (by providing the random seed, etc.). They can try several runs to produce each submitted solution (each with the allowed computer time), but they must be able to repeat the run for any solution submitted.
8. Participants should submit for each instance (early and late) the best score found by their algorithm in the specified computer time, by uploading it onto the competition web site. Participants should also submit a concise and clear description of their algorithm, so that in principle others can reproduce it.
9. Classification will be based on the scores provided. The actual list will be based on the *ranks* of the solvers on each single instance. Ranks will be based hierarchically on hard constraint violations and scores on the soft ones. The average of the ranks will produce the final place-list.
10. Based on the place-list a set of top solvers, the *finalists*, will be asked to provide the executable that will be run and tested by the organisers. The finalists' solver will be rerun by the organisers on all instances (including the hidden ones).
11. In some circumstances, finalists may be required to show source code to the organisers. This is simply to check that they have stuck to the rules.
12. Finalists' place-list will be again based on the ranks on each single instance for a set of trials on the hidden instances.

3 Problem formulation and Instances

The Curriculum-based timetabling problem consists of the weekly scheduling of the lectures for several university courses within a given number of rooms and time periods, where conflicts between courses are set according to the curricula published by the University and not on the basis of enrolment data. This formulation applies to University of Udine (Italy) and to many Italian and indeed

International Universities, although it is slightly simplified with respect to the real problem to maintain a certain level of generality.

The problem consists of the following entities:

Days, Timeslots, and Periods. We are given a number of *teaching days* in the week (typically 5 or 6). Each day is split in a fixed number of *timeslots*, which is equal for all days. A *period* is a pair composed by a day and a timeslot. The total number of scheduling periods is the product of the days times the day timeslots.

Courses and Teachers. Each course consists of a fixed *number of lectures* to be scheduled in distinct periods, it is attended by given *number of students*, and is taught by a *teacher*. For each course there is a minimum number of days that the lectures of the course should be spread in, moreover there are some periods in which the course cannot be scheduled.

Rooms. Each *room* has a *capacity*, expressed in terms of number of available seats. All rooms are equally suitable for all courses (if large enough).

Curricula. A *curriculum* is a group of courses such that any pair of courses in the group have students in common. Based on curricula, we have the *conflicts* between courses and other soft constraints.

The solution of the problem is an assignment of a period (day and timeslot) and a room to all lectures of each course.

3.1 Hard constraints

Lectures: All lectures of a course must be scheduled, and they must be assigned to distinct periods.

RoomOccupancy: Two lectures cannot take place in the same room in the same period.

Conflicts: Lectures of courses in the same curriculum *or taught by the same teacher* must be all scheduled in different periods.

Availabilities: If the teacher of the course is not available to teach that course at a given period, then no lectures of the course can be scheduled at that period.

3.2 Soft constraints

RoomCapacity: For each lecture, the number of students that attend the course must be less or equal than the number of seats of all the rooms that host its lectures.*Each student above the capacity counts as 1 point of penalty.*

MinimumWorkingDays: The lectures of each course must be spread into a minimum number of days. *Each day below the minimum counts as 5 points of penalty.*

CurriculumCompactness: Lectures belonging to a curriculum should be adjacent to each other (i.e., in consecutive periods). For a given curriculum we account for a violation every time there is one lecture not adjacent to any other lecture within the same day. *Each isolated lecture in a curriculum counts as 2 points of penalty.*

RoomStability: All lectures of a course should be given in the same room. *Each distinct room used for the lectures of a course, but the first, counts as 1 point of penalty.*

4 Instances, File Formats, and Validation

There will be 21 instances available: 7 for each set (early, late, and hidden). All instances are real data coming from the University of Udine. For all instances there exists at least one feasible solution (no hard constraint violations), but at present it is not known which is the optimal value for the soft constraints.

In order to model cases in which the number of timeslots is not the same for all days (e.g. Saturday afternoon free), there might be periods unavailable for all courses. For all instances there cannot be two curricula composed by exactly the same courses.

Each instance comes in a single file, containing a file header and four sections: courses, rooms, curricula, and constraints. The header provides all scalar values and each section provides the arrays for that specific aspect of the problem. The exact format is described in the web site.

The output also must be provided in a single file such that each line represents the assignment to one lecture in the following format (lines can be in any order).

We provide the C++ source code of a solution validator, so that the participant can compile it themselves at their machine and also inspect the code. In case it is necessary, executables for various platforms could be provided on request.

The validator takes two command-line arguments: the input file and the output file and it produces on the standard output the evaluation of the solution along with the detailed description of all violations (hard and soft). If the output file is not formatted correctly, the validator produces an error message on the standard error and aborts. Conversely, the input is assumed always correct. An input file validator, in case the participants want to create new instances, is available and can be provided upon request.

5 Discussion

As already mentioned, the description proposed here is preliminary due to the timing of the competition. More details will be added when available to the public.

We believe that the most important advances w.r.t. the previous competition is the employment of both more realistic problem formulations and data coming from the real world.

In addition, in ITC-2002 only feasible solutions were accepted and it was purposely rather simple to produce a feasible one for all instances. This time, participants that reach only infeasible solutions for some instances can submit their solution, although all of them have a feasible one.

In order to compare different solvers in cases of unfeasible solutions for some instances, we use an evaluation based on ranking of solutions on each instance, rather than on the actual scores (which might be incomparable). Due to this scoring based on rankings, an infeasible solution on an instance does not necessarily prejudice the overall performance.

Finally, in ITC-2002 the ranking was fully based on the solution provided by the participants. In case of stochastic solvers, this CPU time was to grant the maximum time *for each single trial*. Therefore, the participants could take advantage of what we call the *Mongolian Horde* approach (in [6]): "Run as many trials as you can and report only the best of all of them". In ITC-2007, the re-running of finalist solvers on organisers machine (with new seeds) and the use of hidden instances should be able to provide against this approach. It is worth mentioning that in order to provide against the excessive use of the Mongolian Horde approach, the competition organisers tested the best few algorithms also on unseen instances, and indeed the results were found to be broadly in-line with the known instances.

References

1. M. Chiarandini, M. Birattari, K. Socha, and O. Rossi-Doria. An effective hybrid approach for the university course timetabling problem. *Journal of Scheduling*, 9(5):403–432, 2006.
2. Luca Di Gaspero and Andrea Schaerf. Neighborhood portfolio approach for local search applied to timetabling problems. *Journal of Mathematical Modeling and Algorithms*, 5(1):65–89, 2006. DOI: 10.1007/s10852-005-9032-z.
3. Philipp Kostuch. The university course timetabling problem with a three-phase approach. In Edmund Burke and Michael Trick, editors, *Proc. of the 5th Int. Conf. on the Practice and Theory of Automated Timetabling (PATAT-2004), selected papers*, volume 3616 of *Lecture Notes in Computer Science*, pages 109–125, Berlin-Heidelberg, 2005. Springer-Verlag.
4. Barry McCollum. University timetabling: Bridging the gap between research and practice (invited paper). In E. Burke and H. Rudová, editors, *Proc. of the 6th Int. Conf. on the Practice and Theory of Automated Timetabling (PATAT-2006)*, pages 15–35, Brno, The Czech Republic, 2006.
5. Andrea Schaerf. A survey of automated timetabling. *Artificial Intelligence Review*, 13(2):87–127, 1999.
6. Andrea Schaerf and Luca Di Gaspero. Measurability and reproducibility in timetabling research: State-of-the-art and discussion (invited paper). In E. Burke and H. Rudová, editors, *Proc. of the 6th Int. Conf. on the Practice and Theory of Automated Timetabling (PATAT-2006)*, pages 53–62, Brno, The Czech Republic, 2006.