Modelling and Solving the Italian Examination Timetabling Problem using Tabu Search

Andrea Zampieri and Andrea Schaerf

Dipartimento di Ingegneria Elettrica, Gestionale e Meccanica
Università di Udine
via delle Scienze 208, I-33100, Udine, Italy

1 Introduction

The Examination Timetabling problem (ETTP) regards the scheduling for the exams of a set of university courses, avoiding the overlap of exams of courses having common students, and spreading the exams for the students as much as possible.

Carter *et al* [3] provide a set of formulations for ETTP, which differ from each other on some components of the objective function, along with a set of benchmark instances [2]. Formulations and benchmarks by Carter have stimulated a large body of research, so that many researchers (see, e.g., [1,5,4]) have adopted one of the formulations of Carter (or a variant of them), tested their algorithms on the benchmarks, and also added new instances. At present, all instances and the corresponding best results are published on the Web [8].

Unfortunately though, the real-world formulation that applies to our institution and, up to our knowledge, to most Italian universities, differs substantially from Carter's and similar ones. Therefore, in order to solve our practical problem, we have been forced to model and solve the specific situation of our university almost from scratch.

In this paper, we present an ongoing research on the development of a solution software for the ETTP at the school of Engineering of the University of Udine. In details, we present the problem modelling, the structure of the real instances, and the solution algorithm, which is based on Tabu Search [7] and on the interleaving of different neighbourhood relations, in the spirit of [6].

2 Italian Examination Timetabling Problem

An examination session takes place after each teaching term and it is composed by a set of w weeks, and each week is divide in 10 periods of half-day length each (Mon to Fri).

For each session, all courses of the university are split into two sets: courses given in the term preceding the session, called *current courses*, and courses given in a different term, called *past courses*. For current courses there must be *two* examinations in the session, whereas past ones must have just *one* examination.

The examination of each course (either current or past) is of one of the following types:

- Single test: the exam takes place in a single period
- Long test: the exam takes place in the two periods of one single day
- Double test: the exam takes place in two separated tests (typically written and oral), with a predefined minimum distance between them

There are groups of courses, called *curricula*, that have students in common. Exams of courses in the same curriculum should be given in different periods and spread as much as possible (as explained below).

There are also groups of courses, called *clusters*, such that their teachers want to have the examinations in the same period. Obviously, courses in the same curriculum cannot be also in the same cluster.

Exam takes place in the *classrooms*, which are split into three categories based on the size: small, medium, and large. Each exam, or cluster of exams, requires for each test a number of rooms for a given type. Classrooms may also have special equipments, so that some exams may take place only in specific classrooms.

Teachers might declare both their unavailability for a (maximum) number of periods and at most three (ordered) *preferred periods*. Exams cannot be scheduled when the teacher is unavailable and should be scheduled in the most preferred periods. If no preference is declared, all available periods are considered at the same level of preference.

We search for the assignment coherent with the rules for current and past exams, and with the exam types, such that the following *hard* constraints are satisfied, and the violations of the following *soft* ones are minimised.

1. **Strong Conflicts (hard):** Exams of *current* courses in the same curriculum must be all scheduled in different periods.
2. **Room Occupancy (hard):** Two distinct exams (not in the same cluster) cannot take place in the same room in the same period.
3. **Room Capacity and Type (hard):** The number of rooms must be equal to the request. The size must be at least equal to the request. If an exam requires a special equipment, it must be assigned only rooms with that equipment.
4. **Clusters (hard):** Exams belonging to the same cluster must be scheduled in the same period. If two exams in the same cluster have different exam type, then only the first test has to be scheduled together.
5. **Availabilities (hard):** An exam cannot be scheduled when the teacher is unavailable.
6. **Weak Conflicts (soft):** Exams of courses in the same curriculum must be all scheduled in different periods.
7. **Day Conflicts (soft):** Exams of *current* courses in the same curriculum must be all scheduled in different days.
8. **Same-exam distance (soft):** The two exams of the same current course must be scheduled with a given minimum distance.
9. **Distance (soft):** Exams of current courses in the same curriculum must be scheduled in periods with a given minimum distance.
10. **Preferences (soft):** If a teacher has expressed some preferences, exams scheduled in the second, third or not preferred periods are penalised gradually according to a fixed weight pattern.

The objective function is the weighted sum of the soft constraints listed above. Weight are adjusted according to experience and discussions with the dean of the school.

3 Tabu Search for Examination Timetabling

We make use of a tabu search (TS) algorithm. In order to apply TS to our problem we have to define several features. We first illustrate the search space and the procedure for computing the initial state. Then, we define the neighbourhood structure, and finally we describe the guiding search procedure.

Search Space: For each single exam we define one variable that takes the value of the period the exam takes place (or its first test). Constraint types 4 and 5 are always satisfied, the others are included in the cost function (with a high weight) of the search procedure.

Room-related hard constraints (types 2 and 3) are checked and evaluated in each state, but the actual rooms are assigned only in a post-processing step.

Initial Solution Selection: The initial solution is selected assigning each exam at the most preferred period. If there are no preferences, the exam is assigned at random, with constraints types 4 and 5 satisfied.

Neighbourhood Relation: We consider two neighbourhood relations. The first one, called change, assigns a new period to an exam (or cluster). The second one, called swap, exchanges the periods assigned to two exams (or cluster). Moves that violates constraints 5 are not considered.

Search Techniques: Our algorithm interleaves different *runners* (as defined in [6]): Runners are invoked sequentially and each one starts from the best state obtained from the previous one. The overall process stops when a full round of all of them does not find any improvement. Each single runner stops when it does not improve the current best solution for a given number of iterations (called *max idle iterations*).

We experimented with various solvers that differ from each other on the number and type of runners they use (see below). The base runners are two tabu search using the two neighbourhoods proposed above and a hill climbing using the change neighbourhood; we name them TS(ch), TS(sw) and HC(ch) respectively.

4 Experimental Results

The available dataset is composed by three real-world instances coming from past sessions in our university. The main features of the instances are summarised in Table 1. In details, we show the total number of exams to be scheduled (considering clusters as single courses), the number of periods and rooms, and the percentage of conflicting courses (all types of conflicts), and the percentage of request of rooms (with respect to rooms × periods). All data will be made available to the community for comparison through a web page shortly.

Instance	Total Exams	Periods	Rooms	% Total Conflicts	% Room Occupation
1	293	40	17	8.73%	40.85%
2	434	40	17	12.32%	69.29%
3	332	40	18	8.41%	47.76%

Table 1. Features of the Instances

Table 2 shows a comparison of the results of the most promising solvers. For each solver, we run 10 trials for each instance and we report the minimum and the average cost. All trials have been granted 1 minute of computing time.

		Instance 1		Instance 2		Instance 3	
	Solver	best	avg	best	avg	best	avg
1	TS(ch)	3098	3350.0	2806	2873.0	3794	4523.4
2	TS(ch) + TS(sw)	2927	3073.2	2573	2596.8	3737	4340.6
3	HC(ch) + TS(ch)	3225	3481.8	2624	2746.0	4166	4658.8
4	HC(ch) + TS(ch) + TS(sw)	2911	3180.6	2502	2569.2	3858	4201.4

Table 2. Comparison of solvers

Table 2 shows that the best results are obtained by solvers 2 and 4, which have the composition of the two TS (without and with HC, respectively). The difference between these two is small, in favour of solver 4.

We conclude showing the results obtained using the *restricted* search space, which is composed by allowing for each exam only the three periods preferred by the teachers (all periods if no preference was expressed). This is the search space underlying the manual construction of the solution used by the university before the development of our software.

Instance	Best	Avg
1	9963	11867.0
2	16045	16613.0
3	19069	20010.6

Table 3. Results using the restricted search space

Table 3 shows the best results obtained using such a search space. It is easy to see that the results are much worse than those presented in Table 2. The fact that there would be a loss of quality was obvious, but the numerical difference (almost one order of magnitude) highlighted in Table 3 are somewhat surprising. This result shows that the use of the larger search space is necessary, but this was beyond the ability of the human scheduler.

References

1. E. Burke and J. Newall. A multi-stage evolutionary algorithm for the timetable problem. *IEEE Transactions on Evolutionary Computation*, 3(1):63–74, 1999.
2. M. W. Carter. Carter's test data. URL: `ftp://ftp.mie.utoronto.ca/pub/carter/testprob/`, 2005. Viewed: June 1, 2006, Updated: June 7, 2005.
3. M. W. Carter, G. Laporte, and S. Y. Lee. Examination timetabling: Algorithmic strategies and applications. *Journal of the Operational Research Society*, 74:373–383, 1996.
4. S. Casey and J. Thompson. Grasping the examination scheduling problem. In Edmund Burke and Patrick De Causmaecker, editors, *Proc. of the 4th Int. Conf. on the Practice and Theory of Automated Timetabling (PATAT-2002), selected papers*, volume 2740 of *Lecture Notes in Computer Science*, pages 232–244, Berlin-Heidelberg, 2003. Springer-Verlag.
5. Luca Di Gaspero and Andrea Schaerf. Tabu search techniques for examination timetabling. In E. Burke and W. Erben, editors, *Proc. of the 3rd Int. Conf. on the Practice and Theory of Automated Timetabling (PATAT-2000), selected papers*, volume 2079 of *Lecture Notes in Computer Science*, pages 104–117. Springer-Verlag, Berlin-Heidelberg, 2001.
6. 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.
7. Fred Glover and Manuel Laguna. *Tabu search.* Kluwer Academic Publishers, 1997.
8. Liam Merlot. Public exam timetabling data sets. URL: `http://www.or.ms.unimelb.edu.au/timetabling`, 2005. Viewed: June 1, 2006, Updated: October 13, 2003.