Post Enrolment based Course Timetabling: A Description of the Problem Model used for Track Two of the Second International Timetabling Competition

Rhydian Lewis[1] **Ben Paechter**[2] **Barry McCollum**[3]

[1]Cardiff Business School, Pryfysgol Caerdydd/Cardiff University,

Cardiff, Wales, CF10 3EU.

Email: lewisr9@cf.ac.uk

[2]Centre for Emergent Computing, Napier University,

Edinburgh, Scotland, EH10 5DT.

Email: b.paechter@napier.ac.uk

[3]SARC Building, School of Electronics, Electrical Engineering and Computer Science, Queens University,

Belfast, Northern Ireland.

Email: b.mccollum@qub.ac.uk

Cardiff Working Papers in Accounting and Finance A2007-3. Prifysgol Caerdydd/ Cardiff University, Wales. ISSN: 1750-6658, v 1.0.

August 2007

Abstract: In this paper we give a detailed description of the problem model used in track-two of the second International Timetabling Competition, 2007-2008 (www.cs.qub.ac.uk/itc2007/). This model is an extension of that used in the first timetabling competition, and we discuss the rationales behind these extensions. We also describe in detail the criteria that are used for judging solution quality and discuss other issues that are related to this. Finally we go over some of the strengths and limitations of the model. This paper can be regarded as the official documentation for track-two of the competition.

1 Introduction

The timetabling of events (such as lectures, tutorials, and seminars) at universities in order to meet the demands of its users is often a difficult problem to solve effectively. As well as wanting a timetable that can actually be used by the institution, users will also want a timetable that is “nice” to use and which doesn’t overburden the people who will have to base their days’ activities around it. Timetabling is also a very idiosyncratic problem that can vary between different countries, different universities, and even different departments. From a computer-

science perspective, it is therefore a problem that is quite difficult to study in a general way.

The Second International Timetabling Competition (www.cs.qub.ac.uk/itc2007/) has been organised to allow researchers from various fields to compare and contrast timetabling algorithms using a common set of benchmark instances in an accurate and fair way. The competition has been split into three tracks, each of which deals with a different type of university timetabling problem; namely exam timetabling, post enrolment-based course timetabling, and curriculum-based timetabling. The main rules of the competition, which are universal to all three tracks, are described in detail on the competition website.

The timetabling problem-version that is described in this document is the Post Enrolment-based Course Timetabling Problem used in track-two of the competition. This particular model is intended to simulate the real-world situation where students are given a choice of lectures that they wish to attend, and the timetable is then constructed according to these choices (that is, the timetable is to be constructed *after* students have selected which lectures they wish to attend). Our intention in this document is to describe this problem in detail, to outline the judging criteria and other related rules that are used with this model, and to discuss its general merits and limitations.

The Post Enrolment-based Course Timetabling Problem model is based on the model that was used in the first international timetabling competition (http://www.idsia.ch/Files/ttcomp2002/), which was run in 2003 in conjunction with PATAT and the Metaheuristics Network. It should be noted that the problem model used in the first competition has been given various names in the literature including the “Class Timetabling Problem”, the “Event Timetabling Problem”, the “Class assignment Problem”, and the “University Course Timetabling Problem”. Readers who are interested in researching some of the work conducted with the problem model used in the first competition are directed to the work of Lewis (2006, 2007), Kostuch (2005), Chiarandini *et al*. (2003), Socha *et al*. (2002), and Rossi-Doria *et al*. (2002). Various pieces of useful information can also be found on the original competition’s webpage.

2 Problem Background

In the original timetabling competition, a problem model was used in which a number of “events” had to be scheduled into rooms and “timeslots” in accordance with a number of constraints. These constraints can be divided into two classes: the *hard* constraints and the *soft* constraints. The former are mandatory in their satisfaction and reflect constraints that need to be satisfied in order for the timetable to be useable; the latter are those that are to be satisfied only *if possible* and are intended to make a timetable “nice” for the people who were supposed to use it.

One important feature of the original competition model was the way in which the quality of the entrants’ solutions was measured. It was decided by the competition

organisers beforehand that timetables would only be judged by calculating the number of soft constraint violations within the proposed solution. In fact, algorithms were only eligible to enter the competition if feasible timetables could be produced within the time limit. One reason for this was to avoid the problem of deciding how to compare two solutions with different numbers of broken hard constraints *and* different numbers of broken soft constraints. Consequently, the problem instances that were used in this competition were specially constructed so that the hard constraints in each case were generally quite easy to satisfy.

One effect of this judging criterion was that the majority of ideas generated in the first competition were to do with the satisfaction of the soft constraints. That is, many of the algorithms that were entered would operate by quickly satisfying the hard constraints of a particular problem instance and would then devote the majority of their time-and-effort in attempting to satisfy the soft constraints of the problem (while not re-violating any of the hard constraints in the process). Some of these algorithms were very effective and added valuable knowledge to the field. However, in many real-world timetabling situations, satisfying the hard constraints of a given problem may not always be so easy. Therefore, in the second competition, we have chosen to use problems where this task is not so straightforward.

As mentioned, the problem model that is used in Track 2 of the Second International Timetabling Competition is an extension of the problem model used in the first competition. However, in this case, extra constraints have also been added to the model to move further in the direction of real-world timetabling. This has been achieved by adding two extra hard constraint types, which we will now outline.

3 Problem Description

The Post Enrolment-based timetabling model used in track-two of the Second International Timetabling Competition can be defined as follows. To begin with, each problem consists of the following information (note that the exact layout of this information in each problem instance file is given on the competition website at www.cs.qub.ac.uk/itc2007/):

- A set of *n* events that are to be scheduled into 45 timeslots (5 days of 9 hours each);
- A set of *r* rooms, each which has a specific seating capacity, in which the events take place;
- A set of *f* room-features that are *satisfied* by rooms and which are *required* by events.
- A set of *s* students who *attend* various different combinations of events;
- A set of *available* timeslots for each of the *n* events (i.e. not all events will be available in all timeslots);

- A set of *precedence* requirements that state that certain events should occur before certain others.

The aim is to try and insert each of the *n* events into the timetable (that is, assign each of the *n* events to one of the *r* rooms and one of the 45 timeslots) while obeying the following five hard constraints:

1) No student should be required to attend more than one event at the same time;
2) In each case the room should be big enough for all the attending students and should satisfy all of the features required by the event;
3) Only one event is put into each room in any timeslot;
4) Events should only be assigned to timeslots that are pre-defined as "available" for those events;
5) Where specified, events should be scheduled to occur in the correct order in the week.

Note that hard constraints 1), 2), and 3) above are exactly the same as the hard constraints that were used in the first competition. Constraints 4) and 5), meanwhile, are new additions to the model.

Since it is now unrealistic to expect all algorithms to satisfy all of the hard constraints within the given time limit, we had to address the problem of how to deal with infeasible timetables. Our solution was to say that that solutions submitted still had to be free of hard constraint violations, but that this could be achieved by leaving some events out of the timetable or "unplaced". We will return to this topic in Section 4 below.

At this point it is useful for us to define some terminology:

- A ***valid*** timetable is one in which there are no occurrences of any hard constraint violations, but some of the events have been left to one side unplaced.
- A ***feasible*** timetable is one in which there are no occurrences of any hard constraint violations, and *all* of the events are present in the timetable.

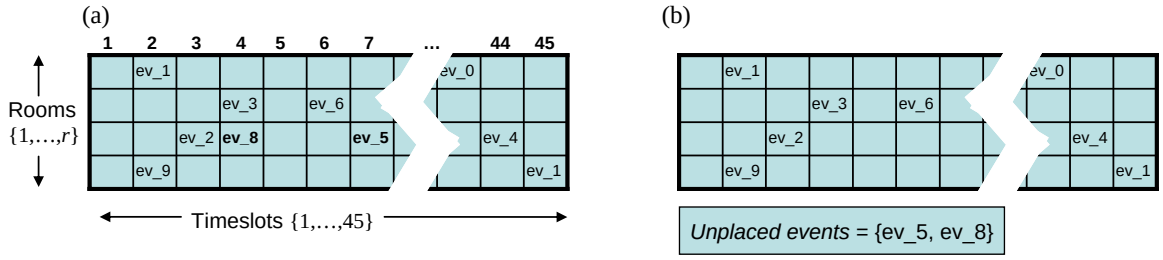
For clarity, these concepts are illustrated in fig. 1.

In addition, to the five hard constraints that are given above, in this problem model we are also interested in satisfying a number of *soft constraints*. These are as follows:

1) Students should not be scheduled to attend an event in the last timeslot of a day (that is, timeslots 9, 18, 27, 36, or 45);
2) Students should not have to attend three (or more) events in successive timeslots occurring in the same day;
3) Students should not be required to attend only one event in a particular day.

Note that these three soft constraints are the same as those used in the first competition.

Solutions to an instance of this problem are to be written to a file in a very simple text format that is described on the competition web-page. The competition organisers have made available a program that checks these solution files against the given problem file and outputs a summary of the constraint violations etc. The source code for this program (written in C++) is also available on the competition website.



Imagine in this case that events 5 and 8 (ev_5, ev_8) are causing a violation of some hard constraint. In this case this timetable is **invalid**.

In this case, events 5 and 8 have been removed from the timetable. The timetable is incomplete, but it is **valid**

Fig. 1: Example of an invalid and valid timetable according to the competition criteria

4 Solution Evaluation

In this section we will now describe the rules that are used in this competition track for measuring a timetabling solution's quality.

To start with, all submitted solutions must be *valid* – otherwise they are disqualified from the competition. Recall, however, that it *is* permissible for some of the events to be left unplaced. If this is the case we can use these unplaced events in order to calculate a *Distance to Feasibility* measure. This is calculated by identifying the number of students that are required to attend each of the unplaced events and then simply adding these values together. Thus if, for example, a solution has three events that are unplaced, and the number of students attending each of these is 12, 8, and 5, then the Distance to Feasibility is simply (12 + 8 + 5) = 25. Note that a *feasible* timetable, by definition, has a Distance to Feasibility of zero.

Having measured the Distance to Feasibility, the number of soft constraint violations is then considered. This is calculated in the following way (which is identical to the method used in the first competition):

- Count the number of occurrences of a student having just one class on a day (count 2 if a student has two days with only one class, etc.).
- 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.

The *Soft Cost* of the timetable is simply the total of these three values.

From the above descriptions we can see that a valid timetable's quality is therefore reflected by a pair of values: (1) the Distance to Feasibility, and (2) the Soft Cost. In order to directly compare two solutions (and judge which one is best), we then use the following sequential procedure:

First, we examine the solutions' Distances to Feasibility. The solution with the lowest value for this is then deemed the winner. However, if the two solutions are equal in this respect, we then look at the number of soft constraint violations contained in each of the solutions. The winner is then judged to be the one that has the lowest Soft Cost.

4.1 Comparing Timetables: Some Practical Issues

In the paragraphs above, we have explained the method that is used for calculating a valid timetable's Distance to Feasibility, and the scheme that is used for comparing the quality of two different solutions. We have chosen this particular method as this is the sort of thing that might be done in real-world timetabling. There are other methods that we could have chosen for doing this, each which will have advantages and disadvantages. What is important is that we have a fair method of comparing the solutions that have been produced by different timetabling algorithms – a method that can be understood by competitors in advance.

We are aware that many algorithms in the literature do not follow the strategy of leaving events unplaced. For example, some algorithms will insert *all* of the events into the timetable and then set about trying to eliminate as many of the hard constraint violations as possible. (In such algorithms a typical "Distance to Feasibility" measure will be some value that reflects the total number of hard constraint violations in the timetable. See, for example, the work of Schaerf *et al*. (1999).) This type of method may be preferable in some practical situations since it will allow the possibility of scheduling some events despite the fact that they are breaking some hard constraints. (In this case, such an approach effectively makes the hard constraints a type of soft constraint).

If entrants choose to implement an algorithm that follows this kind of strategy, then they will also need to implement a simple procedure, possibly to be used at the end of the run, which removes certain events from the timetable in order to eliminate any hard constraint violations that might be occurring. In our experience, such procedures are easy to implement in practice. Our choice of evaluation method does not reflect any view of the relative merits of the different algorithm varieties.

Finally, it is worth reiterating that, according to this measure, it is not the number of unplaced events that are important for this measure, rather it is the *number of*

students within these unplaced events. Again, this reflects what we believe to be a real-world situation, where we are trying to satisfy as many people's needs as possible within the timetable.

5 Model Limitations

In the previous sections we have noted that the new timetabling model has a number of added features that are intended to move this problem towards those that we might expect to encounter in the real world (e.g. McCollum 2007). However, in order to maintain a degree of generality in these studies, and also to avoid overwhelming the competition entrants with a huge set of constraints, we have also avoided imposing a number of real world features on this problem. However, for completeness, and also to make the reader aware of other characteristics that we might expect to encounter in real world timetabling, in this section we will now identify some of these.

The first types of constraint that we may wish to consider are those that are concerned with the *relative positioning* of events within the timetable. We have already addressed this aspect of timetabling to a certain degree with the imposition of the precedence constraints (i.e. constraint 5) in Section 3. However, there are, of course, also a number of other constraints of this type that could be encountered in practice. The following three examples are typical but not exhaustive:

- **Inter-site travel times:** in some practical cases, a university might be split across a number of campuses, and students and staff may require some commuting-time in order to travel from one site to another. Thus, if two events *i* and *j* have common students, but need to take place in different sites, then the constraint "if event *i* is scheduled to occur in timeslot *x*, then event *j* cannot occur in timeslot $x + 1$ if this timeslot is on the same day" might be specified. Other related constraints might give a penalty to changing rooms or corridors unnecessarily.
- **Providing a Lunch-break:** many universities will also want to ensure that all staff and students have the opportunity to eat lunch. Thus constraints such as the following might be imposed: "if a student is attending an event in a 12:00pm timeslot, then he-or-she must not be required to attend and event in a 1:00pm timeslot on the same day, and vice-versa".
- **Relative Timing of Events:** universities may also wish to impose other types of constraint on their timetabling problem such as "events *i* and *j* must be assigned to the same/different timeslots", "events *i* and *j* must take place on the different days", "there must be at a least one day gap between events *i* and *j*", and so on.

In real-world timetabling there are often extra issues concerning rooms. For example:

- **Events without Rooms:** in certain cases some events may not actually require a room, because they may take place outdoors, involve trips to off-site locations, and so on.
- **Room availability:** In some cases, certain rooms might not be available in certain timeslots. This could be caused by, say, the room being used by another faculty, or because the key-holder of the room might not be present at certain times during the week.
- **Room Hierarchies:** in many institutions, a large room may have a number of movable partitions within it, so that the room can be effectively broken up into a number of smaller classrooms. This means that in one timeslot, the resource might be used to house a very large event, while in the next timeslot a number of smaller events might all be scheduled into this same resource.
- **Filling Rooms:** In some cases, the university may have a policy where small events are discouraged from being put into overly large lecture theatres etc.

As well as all of these features, there are also an abundance of different constraints relating to the usability and "friendliness" of a timetable. Such constraints, usually expressed as soft constraints can include:

- **Free days:** in some institutions, it may be considered desirable to allow students and/or staff to have one day a week free from lectures in order to allow time for research etc.;
- **Lecturer Preferences:** There may also be a number of individual requirements from lecturers about the allocation of their teaching hours. Some lecturers, for example, may prefer to do all of their teaching in a single day; others may prefer to have their hours equally distributed throughout the week.

Finally, another timetabling feature that has not currently been considered is the occurrence of variable length events – this could range from allowing double or triple length events to, for example, allowing events to start and finish at ten minute intervals.

Note that, for the reasons stated earlier, all of the problem features and constraints discussed in this section have been deliberately left out of the problem model used in the current timetabling competition. However, some of these could be introduced in future versions of the competition if it were deemed appropriate to do so.

References

Chiarandini, M., Socha, K., Birattari, M., and Rossi-Doria, O. (2003) "An Effective Hybrid Approach for the University Course Timetabling Problem", *Technical Report AIDA-2003-05*, FG Intellektik, FB Informatik, TU, Darmstadt, Germany.

Lewis, R. (2006) “Metaheuristics for University Course Timetabling”. Doctoral Thesis, Napier University, Edinburgh, Scotland. (Available at http://www.cardiff.ac.uk/carbs/ quant/rhyd/rhyd.html)

Lewis, R. (2007) “A Survey of Metaheuristic-based techniques for University Timetabling Problems” *OR Spectrum*, DOI 10.1007/s00291-007-0097-0 (Available at http://www.cardiff.ac.uk/carbs/ quant/rhyd/rhyd.html)

Kostuch, P. (2005) “The University Course Timetabling Problem with a 3-Phase Approach”, in E. Burke M. Trick, (eds.) the Practice and Theory of Automated Timetabling (PATAT) V, *Lecture Notes in Computer Science vol.* 3616, Springer Verlag: Berlin, pp 109 – 125.

McCollum, B. “A Perspective on Bridging the Gap in University Timetabling”, PATAT '06, Proceedings of the 6th International Conference on the Practice and Theory of Automated Timetabling, Brno, August 2006. (Accepted for post conference Springer Lecture Notes in Computer Science Volume)

Rossi-Doria, O., Samples, M., Birattari, M., Chiarandini, M., Knowles, J., Manfrin, M., Mastrolilli, M., Paquete, L., Paechter, B., and Stützle, T. (2002) “A Comparison of the Performance of Different Metaheuristics on the Timetabling Problem”, in E. Burke, and P. De Causmaecker (eds.) the Practice and Theory of Automated Timetabling (PATAT) IV, *Lecture Notes in Computer Science vol.* 2740, Springer Verlag: Berlin, pp 329 – 351.

Schaerf, A. (1999) “Local Search Techniques for Large High-School Timetabling Problems”, *IEEE Transactions on Systems Man. and Cybernetics Part A*, 29(4), pp 368 – 377.

Socha, K., Knowles, J., and Samples, M. (2002) “A MAX-MIN Ant System for the University Course Timetabling Problem”, In M. Dorigo, G. Di Caro, and M. Samples (eds.), Proceedings of Ants 2002 - Third International Workshop on Ant Algorithms (Ants’ 2002), *Lecture Notes in Computer Science vol.* 2463, Springer Verlag: Berlin, pp 1 – 13.