Ann Oper Res (2012) 194:291–315
DOI 10.1007/s10479-011-0997-x

A new model for automated examination timetabling

Barry McCollum · Paul McMullan · Andrew J. Parkes · Edmund K. Burke · Rong Qu

Published online: 6 December 2011

Abstract Automated examination timetabling has been addressed by a wide variety of methodologies and techniques over the last ten years or so. Many of the methods in this broad range of approaches have been evaluated on a collection of benchmark instances provided at the University of Toronto in 1996. Whilst the existence of these datasets has provided an invaluable resource for research into examination timetabling, the instances have significant limitations in terms of their relevance to real-world examination timetabling in modern universities. This paper presents a detailed model which draws upon experiences of implementing examination timetabling systems in universities in Europe, Australasia and America.

This model represents the problem that was presented in the 2nd International Timetabling Competition (ITC2007). In presenting this detailed new model, this paper describes the examination timetabling track introduced as part of the competition. In addition to the model, the datasets used in the competition are also based on current real-world instances introduced by EventMAP Limited. It is hoped that the interest generated as part of the competition will lead to the development, investigation and application of a host of novel and exciting techniques to address this important real-world search domain. Moreover, the mo-

B. McCollum · P. McMullan
School of Computer Science, Queen's University Belfast, University Road, Belfast, N. Ireland, BT7 1NN, UK

B. McCollum
e-mail: b.mccollum@qub.ac.uk

P. McMullan
e-mail: p.p.mcmullan@qub.ac.uk

A.J. Parkes (✉) · E.K. Burke · R. Qu
School of Computer Science, University of Nottingham, Jubilee Campus, Nottingham, NG8 1BB, UK
e-mail: ajp@cs.nott.ac.uk

E.K. Burke
e-mail: ekb@cs.nott.ac.uk

R. Qu
e-mail: rxq@cs.nott.ac.uk

tivating goal of this paper is to close the currently existing gap between theory and practice in examination timetabling by presenting the research community with a rigorous model which represents the complexity of the real-world situation. In this paper we describe the model and its motivations, followed by a full formal definition.

1 Introduction

Building on the success of the First International Timetabling Competition in 2002 (Metaheuristics Network 2003), the second competition (ITC2007)[1] was introduced with the overall aim of attracting researchers to develop and test leading edge techniques within a competitive arena. It also aimed to generate further interest in the research area by providing various formulations of the timetabling problems encountered within educational institutions and therefore based on a real-world perspective. Particular emphasis was placed on real-world scenarios with the objective of encouraging the production of techniques which have the potential to solve practical instances of the problem. The competition therefore had, and will continue to have, an important part to play in bridging the current gap which exists between research and practice in this area (McCollum 2007).

In order to tackle the main variations which exist within the practical area of educational timetabling, the competition was divided into three sections or tracks.

1. Examination Timetabling (Exam TT).
2. Post-Enrolment Course Timetabling (Post-Enroll CTT).
3. Curriculum-Based Course Timetabling (Curriculum CTT).

The competition introduced three tracks along with associated benchmark datasets. From a research perspective this division is important in that it provides a framework to capture the main types of educational timetabling research currently taking place within the academic community. Although the tracks all fall under the general 'umbrella' of educational timetabling, the identified problem areas have significant differences which are discussed in detail at the Competition website and in the associated article (McCollum et al. 2010). In addition, technical reports for all areas are available from the official website and can be found under each track (McCollum et al. 2007; Lewis et al. 2007; Di Gaspero et al. 2007).

In this paper, we present the real-world model that was used in the Examination Timetabling Track. The information presented here is self-contained, but can also be regarded as a detailed and rigorous description of the problem presented in that track and incorporates and extends the content on the ITC2007 website. In addition to the presentation of the model, we also give a description of the competition's hierarchical solution evaluation methods.

1.1 The examination timetabling problem: modelling

Previous research has concentrated on the Toronto benchmark dataset[2] introduced by Carter et al. (1996) at the University of Toronto. These benchmarks and the various issues associated with them are discussed in more detail in Qu et al. (2009). However, the Toronto dataset uses only a very limited model, and so the complexity of modelling of timetabling problems

[1] http://www.cs.qub.ac.uk/itc2007/.

[2] ftp://ftp.mie.utoronto.ca/pub/carter/testprob/.

continues to pose a series of important and challenging issues in the timetabling research area (McCollum 2007).

There is some research in the literature on building general timetabling languages and tools in an attempt to model real world instances of the problem. Tsang et al. (1999) developed a high level language to specify examination timetabling problems as constraint satisfaction problems. Di Gaspero and Schaerf (2001) built a software tool called EASYLOCAL++ for the easy implementation of local search algorithms on general timetabling problems.

De Werra et al. (2002) presented a simple model and some possible extensions for class-teacher timetabling problems. The computational complexity of these problems was also studied, showing some variants of the problem as NP-complete. However, significant further work is required to address these and related issues. Moreover, it is needed to provide fundamental support for a better understanding and development of real-world examination timetabling research. There is a very wide variety of institutional practices, for example, Al-Yakoob et al. (2010) gives an integer programming model of an examination timetabling problem that includes "idiosyncrasies of the problem related to gender-based policies and having multiple exam centers."

Generally, these modelling issues have not been widely discussed over the last ten years, and given the wide variety of institutional practice it is not surprising that there are no universal complete models. However, the point of this paper is to provide a model that is substantially more general and reusable than existing frameworks. Our intention is that it will further improve the current development and justify clear and meaningful scientific comparisons.

1.2 The examination timetabling problem: algorithms

The purpose of this paper is not to present algorithms however for context and completeness we briefly discuss some of the existing algorithms. The last ten years have seen a significant amount of research; for a survey, see Qu et al. (2009). Naturally, meta-heuristics have been widely studied, for example, Tabu Search (Di Gaspero 2002; White and Xie 2001; Paquete and Stützle 2002), Simulated Annealing (Burke et al. 2004; Merlot et al. 2003; Thompson and Dowsland 1998), Genetic Algorithms (Ross et al. 1996; Wong et al. 2002), Memetic Algorithms (Burke et al. 1996), Ant Algorithms (Dowsland and Thompson 2005; Azimi 2005), and Multi-objective Evolutionary Algorithms (Mumford 2007). There is a substantial body of work where hybridisations between meta-heuristics are studied. Indeed, these hybridisations are currently the most effective approaches on the standard Toronto benchmarks. This includes the effective integration of early timetabling techniques such as graph heuristics (Burke and Newall 2004; Burke et al. 2005, 2006, 2007) and constraint-based techniques (Merlot et al. 2003; David 1998; Duong and Lam 2004).

Along with these main themes of algorithm research there are also a number of new trends including more effective design of neighborhood structures (e.g. variable neighborhood search for timetabling, Burke et al. 2010; Ahmadi et al. 2003). Flexibility of search is thus improved to tackle more complex problems with a wider range of constraints. Some research motivated by the objective of raising the generality of timetabling approaches has also obtained promising results, hyper-heuristics (Burke et al. 2003) being one of the areas that is attracting significant research attention (Burke et al. 2002, 2005, 2006, 2007, 2010; Ross et al. 2004).

1.3 Overview of the paper

The Examination Timetabling Problem addressed here introduces a practical formulation of the problem which represents the situation as it exists in many modern universities and which will set the agenda for future efforts in the area by establishing new challenging benchmarks which are relevant to the needs of the user community. An additional goal is that the interest generated by the release of this model will lead to the development, investigation and application of a host of novel and exciting techniques not previously tried within this important real-world search domain.

The problem model can be described as post enrollment. That is, students enrolled on particular courses which have associated exams are considered to be enrolled on or 'taking' those exams. Although other approaches to the problem are taken within institutions, this is by far the most common from a practical perspective as well as being the most widely reported model of the problem within the academic literature.

The model presented in this paper not only significantly adds to the research field by the introduction of a more 'real' representation of the problem, but is also associated with real benchmarks. All the datasets used as part of the competition were taken from real institutions (with their permission) and were anonymised so that they could be publicly released for the benefit of the research community.

Section 2 has a non-mathematical general description of the problem together with some motivating discussion. In particular, Sect. 2.2 presents the framework used for evaluating solutions. A detailed description of the model is given in Sect. 3. Section 4 presents a mathematical programming formulation of the problem. Section 5 gives the competition data set, the solvers submitted, and the results and rankings obtained. Section 6 describes some possible modifications and future extensions to the model. Section 7 presents some concluding comments.

2 The model

The fundamental problem involves assigning exams to a given number of periods within a defined examination session while satisfying various hard constraints. As with other areas of timetabling, a feasible solution is one in which all but the soft constraints are satisfied, and the quality is a weighted sum of the numbers of violations.

Compared to previous benchmark models, new and additional information is provided on constraints, resources and the examination session. For example, in terms of hard constraints, room numbers and sizes are provided. In addition, information on the structure, duration and number of individual periods is also given. In terms of soft constraints, much more practical information is provided in terms of how an organisation measures the overall quality of a solution.

We remark that our goal in this paper is to give the semantics of the model, and not to delve into the particular syntax in the instance files; such details can be found on the competition website. In this section we give a general introduction to the model. More details will be provided in the following two sections.

2.1 Problem description

The problem consists of the following:

- An examination session is made of a number of periods over a specified length of time: The number of periods and duration of each period are provided.

- A set of exams that are to be scheduled into periods. (Exam codes are not provided, instead we assume sequential numbering beginning with 0.)
- The set of students enrolled on each individual exam: Each student is enrolled on a number of exams. Students enrolled on an exam are considered to 'take' or 'sit' that examination.
- A set of rooms with individual capacities.
- A definition of the constraints that any feasible solution must satisfy.
- Specific instance-dependent constraints on feasible solutions.
- Soft constraints (i.e. those which are desirable but not essential) are outlined: the violation of these constraints contribute to a penalty which is used to evaluate feasible timetables.
- Details including a 'weighting' of particular soft constraints.

A feasible timetable is one in which all examinations have been assigned to a single period and room, and in which all the following constraints are satisfied:

- No student sits more than one examination at the same time.
- The capacity of individual rooms is not exceeded at any time throughout the examination session.
- Period durations are not exceeded.
- Period-related hard constraints are satisfied (e.g. exam A after exam B).
- Room-related hard constraints are satisfied (e.g. exams A and B must be in different rooms).

Notice that, unlike course timetabling, exams are generally permitted to share rooms.

The soft constraints and their violation penalties are discussed in detail later, but can be outlined as follows;

- Two exams in a row: Avoid making students sit two exams in a row on the same day.
- Two exams in a day: Avoid making students sit two exams on the same day even though in non-adjacent periods. This constraint only becomes important when there are more than two examination periods in the same day.
- Specified spread of examinations: Avoid making students sit more than one exam within a time range (number of periods) specified by the institution. This is often used in order to try to satisfy the preference by students that their exams should not be too close together.
- No mixed duration of examinations within individual periods: Avoid assigning exams of different durations to the same room and period. If the exams are different durations then this can make invigilation more difficult, and students leaving the earlier exams could disturb the remaining students.
- Larger examinations appearing later in the timetable: Avoid 'large' exams appearing in the 'later portion' of the timetable. Both 'large' and 'later portion' are user defined. This can be used to give the markers of the large exams more time for marking after their exam.
- Period-related soft constraints: Avoid specific time periods for specified exams. There can be many motivations for such penalties, such as teachers not being available to answer any questions arising during the exam.
- Room-related soft constraints: Avoid the usage of certain rooms. Again this might have many uses, such as the estates office wanting to avoid opening a room unless it is really needed.

These constraints can effectively be split into two groups: those that are resource specific, and those that have a global setting. Period related and room related constraints are resource specific i.e. settings can be established for each period and each room. This allows control of how resources would be used in constructing a solution. Values for these can be found

after the introduction of periods and rooms in the datasets. All other soft constraints can be set relative to each other. These are referred to as global settings.

Institutions may give weights for the soft constraints differently (relative to one another) in order to produce a solution which is appropriate for their particular needs. This is known as building the 'Institutional Model' and the associated weighting is defined here as the Institutional Model Index. This relative weighting of the soft constraints provides a quality measure of the solution to be built, and tailored to the needs of the target institution. These weights are provided within each benchmark dataset released on the competition webpage.[3]

It should be noted that when formulating a solution, it is common place for an institution to 'play' with various settings of soft constraints in an attempt to produce solutions which they judge satisfactory to all the end users. Indeed, this is why we provided the soft constraint weightings in the data as opposed to the problem definition. In addition, including the weights in the data rather than requiring them to be hard coded into the solver enabled us to set different weightings for each dataset. We hope that this encourages the development of solvers that are robust rather than potentially over-tuned to one particular set of weights for a dataset. Once again, this is motivated by our experience that different institutions do indeed have different weights, and so no one set would be completely useful across a range of problems.

The details provided here significantly add to the model of the problem commonly used within the research arena. Of course, how individuals judge that particular solutions are 'satisfactory' is an interesting open research problem and is currently being tackled in a number of novel ways (for example, see Asmuni et al. 2006).

2.2 Framework for evaluation of solutions

Generally, in real-world applications, not all constraints are of equal importance to the end-user. Often, this is captured by simply separating constraints into "hard" constraints that must be satisfied, and "soft" constraints that can be violated at the cost of contributing a penalty to the objective function. However, such a two-level approach might not always match the requirements of users, so it has been generalised to multiple levels to give "constraint hierarchies" (Borning et al. 1987, 1992). The basic idea of such a hierarchy is that constraints are separated into multiple levels, and constraints of a higher level are always to be preferentially satisfied over those of a lower level. The notion of such hierarchies, also known as "prioritised constraints", has been extensively studied in the area of artificial intelligence and the constraint satisfaction approach to solving combinatorial optimisation problems, see for example (Dubois et al. 1996; Meseguer et al. 2003; Henz et al. 2004; Callennec and Boulic 2004). As an example, from Borning et al. (1987) (p. 225):

> In a constraint hierarchy, the programmer or user can state both *required* and *preferential* constraints (also known as *hard* and *soft* constraints). The required constraints must hold. The system should try to satisfy the preferential constraints if possible, but no error condition arises if it can't. We allow an arbitrary number of levels of preference, each successive level being more weakly preferred than the previous one.

Such work permits a hierarchy within the soft constraints; however, we wanted two levels for those constraints that would normally be regarded as required (hard) in the sense that their violation in a final timetable would indeed an 'error condition'. The models in all

[3] http://www.cs.qub.ac.uk/itc2007/.

three tracks of ITC2007 effectively used three levels of constraints which we denote here as follows:

1. STRICT-HARD: "hard and never relaxable". Constraints that can *never* broken. Any solution violating such constraints cannot be used and is considered worthless.
2. RELAXABLE-HARD: "hard but relaxable during the process of diagnosis and repair of infeasibility". Constraints that ultimately must be satisfied but for which violations can be tolerated *temporarily*. The corresponding solutions can be used during the *process* of selecting resources and creating a timetable, but not in the final timetable.
3. SOFT: "normally broken". The sets of constraints that we prefer to satisfy but we expect that it will *normally* be partially broken, that is, it will generally not be possible to satisfy them all.

The RELAXABLE-HARD and SOFT constraints can be broken, and so it is natural to assign separate penalty scores measuring the extent to which they are broken. Hence each solution satisfying the required constraints is given a score of a pair of numbers, (d, s), with

d: Penalty equal to the number of violated RELAXABLE-HARD constraints. We will informally refer to this as the "*distance to feasibility*" (though it is not necessarily a form of distance in the solution space).
s: Penalty measuring the breakage of the SOFT constraints.

The notion of a hierarchy is that improving the solution at any level takes precedence over improving it at lower levels. In terms of the scores this means that comparison is first in terms of the value of d, only if the d-scores are equal then the s-scores are compared. That is, the (total) ordering is defined by

$$(d_1, s_1) > (d_2, s_2) \quad \text{if and only if} \quad d_1 > d_2 \text{ or } (d_1 = d_2 \text{ and } s_1 > s_2). \tag{1}$$

By a "feasible solution" we mean one that satisfies all of STRICT-HARD and RELAXABLE-HARD constraints. Thus all feasible solutions have $d = 0$ and the comparison of their scores reduces to the standard comparison of their soft violation scores, s.

Of course, the ordering provided by (1) is also equivalent to taking a single objective score $Md + s$ with M being any weight that is large enough such that any improvements in d always improve the overall value independent of the change in s. (Though note that large weights that can risk problems of numerical overflow and stability, and so maybe not be the best implementation method.) This method of implementation would correspond to using two levels of SOFT constraints, typically called strong and weak in the constraints literature though not mathematical programming. One could then regard the three levels as 'hard', 'highly-penalised soft', and 'normal soft'. However, we believe that the terminology of RELAXABLE-HARD as opposed to 'highly-penalised soft' better captures the intent for real-world practice, as opposed to the method used to implement this intent. To further explain the terminology chosen here: Feasible solutions are generally taken to mean those that can be implemented in practice without any modification or adjustment to the working practices. If only SOFT constraints are violated, then even though some people might be slightly unhappy with parts of the solution, the solution as a whole is still usable, without any special actions being necessary. Then both the STRICT-HARD and RELAXABLE-HARD constraints are indeed hard in the sense that they must be satisfied for the solution to be usable "out of the box". However, real-world timetabling is often a dual process during which decisions are often being made about resources (such as available rooms and timeslots) at the same time as building a timetable. In the case that resources are slightly insufficient then

the problem can easily become infeasible. Rather than the solver simply stating the problem to be infeasible, it is much more helpful to the user to provide information about the reasons for the infeasibility, and so direct how to repair it. A solution to the relaxed problem, in which RELAXABLE-HARD violations are permitted, can still be informative as to how the problem itself can be (minimally) modified so as to restore feasibility. Note the violations are (implicitly) used to drive a change of the model, and so 'in spirit' are different from strong but preferential constraints in which the large penalty is simply accepted. For example, we will see that preventing a student having to take two exams at the same time is only considered to be a RELAXABLE-HARD constraint. The person responsible for the timetabling could use such a solution in order to generate a repair to the model; for example, removing the student from one of the exams and giving a special exam in the immediately following (or preceding) period. Also, in real-world exam timetabling, at the stage of initial planning, infeasibility will typically be met with addition of extra resources such as extra rooms or extra periods. That is, the incumbent timetable would normally restore feasibility by introducing another period or indeed another room and set a high penalty for their use. Typically, infeasibility also tends to arise at later stages in the planning process; for example, if the data changes after creating provisional timetables. The intent of the division of the RELAXABLE-HARD constraints is also similar to methods used to diagnose and repair infeasibility in linear programming, for example, to find an "Irreducible Inconsistent Set" (IIS) (van Loon 1981). Within CPLEX[4] this role is played by the "feasOpt" set of options, and for which constraints can be marked by whether or not they are relaxable. There are then options that allow the user to minimise the number of such relaxable constraints in order to better identify and repair the sources of infeasibility. (Of course, in integer programming, as opposed to linear, programming the task of finding the equivalent of an IIS is generally NP-hard.)

Accordingly, the different practical intentions behind the RELAXABLE-HARD and SOFT constraints meant that it was somewhat clearer to keep them separate, and so for the competition it was decided that all three tracks would have these three levels of constraints. The underlying suggestion is that merely collecting everything into just two levels, hard or soft is perhaps an over-simplification of the real-world, and maybe a richer language with three levels (or more) can aid in communications with end-users. Such issues require a detailed investigation and discussion and as such are beyond the scope of this paper, and will be investigated in future work.

3 The details of the model

In order to describe the new model, as used in the ITC2007 competition, we now present the 'STRICT-HARD', 'RELAXABLE-HARD' and 'SOFT' constraints. In order to specify such a 3-level model one first needs to catalogue the relevant constraints, then we need to make decisions as to which level they should be placed. Potentially, each type of constraint could be used at any level, and so in general we could decide separately where to place every instance of each constraint type. Each constraint instance could be assigned to a level and given a weight. However, we have not (yet) found such generality to be necessary. Soft constraints are easiest to identify because solutions routinely violate them. We are then left with constraints that might well be classified as either strict or relaxable and so define a *distance to feasibility*.

[4]See the User Manual from ftp://ftp.software.ibm.com/.

One of the simplest choices for a *distance to feasibility* would be to allow exams to remain unassigned to a period and room. The *distance to feasibility* could then, for example, be the number of such unscheduled exams, or a sum weighted by the number of students in each examination. (This is essentially the choice taken by the post-enrollment CTT track.) It would have the advantage of being simple, however, it is unrealistic for examinations; of all constraints surely the one that every exam must take place is "strict" rather than being "relaxable."

3.1 The STRICT-HARD constraints

All solutions must, without exception, satisfy the following constraints:

- COMPLETE-ALLOCATION: All exams must actually take place. Scheduling problems are never a good reason to not run an exam!
- NO-PERIOD-SPLITTING: Exams cannot be split between periods.
- NO-ROOM-SPLITTING: Exams cannot be split between rooms.

Observe that these correspond to requiring that all exams are assigned to precisely one period and one room.

3.2 The RELAXABLE-HARD constraints

As already discussed in Sect. 2.2, 'RELAXABLE-HARD' constraints are satisfied in feasible solutions, but can be temporarily relaxed with an associated 'Distance to Feasibility' in order to help diagnose and repair the infeasibility. The choice of such constraints is as follows:

- NO-CONFLICTS: Conflicting exams cannot be assigned to the same period. (As usual, two exams are said to conflict whenever they have some student taking them both.)
- ROOM-OCCUPANCY: For every room and period no more seats are used than are available for that room.
- PERIOD-UTILISATION: No exam assigned to a period uses more time than available for that period.
- PERIOD-RELATED: All of a set of time-ordering requirements between pairs of exams are obeyed. Specifically, for any pair (e_1, e_2) of exams, one can specify any of:
  - 'AFTER': e_1 must take place strictly after e_2,
  - 'EXAM_COINCIDENCE': e_1 must take place at the same time as e_2,
  - 'EXCLUSION': e_1 must not take place at the same time as e_2.
- ROOM-RELATED: Room requirements are obeyed. Specifically, for any exam one can disallow the usual sharing of rooms and specify
  - 'EXCLUSIVE': Exam e must take place in a room on its own.

We emphasise that we have presented a particular choice for which constraints are strict and which are relaxable. This choice is made to represent the views of typical institutions, and was used within the ITC2007 competition. It is particularly convenient that all solutions satisfying the STRICT-HARD constraints have precisely one period and room assigned to each exam, allowing for a simple output format, and this also eases the definition of the other constraints. However, it is quite possible that other choices are also useful for specific institutions. For example, a lack of large rooms might lead some institutions to treat the STRICT-HARD NO-ROOM-SPLITTING as a highly-penalised SOFT constraint. Alternatively, the RELAXABLE-HARD ROOM-OCCUPANCY could be converted to SOFT by allowing a few extra seats when really needed. Some similar potentially useful modifications and extensions to the model are discussed in Sect. 6.

3.3 The evaluation function: soft constraints

Within the model presented here, (and within the competition) feasible solutions are classified based on the satisfaction of the soft constraints. The following provides a description of how each soft constraint is calculated, and also discusses its motivations based on real-world experience. We also give pointers to the appropriate subsections in the mathematical formulation of Sect. 4.

3.3.1 Two exams in a row

This calculation considers the number of occurrences where two examinations are taken by students one straight after another, i.e. back to back, and on the same day. Once this has been established, the number of students are summed and multiplied by the number provided in the 'two in a row' weighting within the 'Institutional Model Index'. Note that if a student has an exam in the last period of one day and another the first period the next day, this does not incur a two in a row penalty. (See Sect. 4.9.1.)

3.3.2 Two exams in a day

In the case where there are three periods or more in a day, the number of occurrences of students having two exams in a day which are not directly adjacent, i.e. not back to back, are calculated. The total number is subsequently multiplied by the 'two in a day' weighting provided within the 'Institutional Model Index'. Therefore, two exams in a day are considered as those which are not adjacent i.e. they have at least a free period between them. This is done to ensure a particular exam placing within a solution does not contribute twice to the overall penalty. For example, if two exams were in adjacent periods in the same day, the penalty would be counted only as part of the 'Two exams in a row penalty'. It should be noted that where the examination session contains days with 2 periods, this component of the penalty, although present for continuity, always is zero and hence superfluous. (See Sect. 4.9.2.)

3.3.3 Period spread

This soft constraint enables an organisation to 'spread' an individual's examinations over a specified number of periods. This can be thought of as an extension of the two soft constraints previously described. Within the 'Institutional Model Index', a figure is provided (with the "PERIODSPREAD" keyword) relating to how many periods the solution should be 'spread' over. The higher this figure, potentially the better the spread of examinations for individual students. In the authors' commercial experience, constructing solutions while changing this setting has led to timetables with which the institution is much more satisfied. If, for example, PERIODSPREAD within the 'Institutional Model Index' is set at 7, for each exam we count all the occurrences of enrolled students who have to sit other exams afterwards but within 7 periods i.e. the desired period spread. This total is added to the overall penalty. It should be noted that the occurrences here will have contributed to the penalty calculated for the 'two exams in a row' and 'two exams in a day' penalties. Although, a single occurrence within the solution is effectively penalised twice, it is often necessary due to, as indicated above, many institutions requiring certain spreads to be minimised as an indication of solution quality. (See Sect. 4.9.3.)

3.3.4 No mixed durations

This applies a penalty to a Room and Period (not Exam) where there are mixed durations. The intention here is to try and ensure that exams occur together which are of equal duration. In calculating this portion of the penalty, the mixed duration component of the 'Institutional Model Index' is calculated by the number of violations detected. (See Sect. 4.9.4.)

3.3.5 Front load

It is often desirable that examinations with the largest numbers of students are timetabled at the beginning of the examination session. In order to take account of this the "FRONTLOAD" expression is introduced. Within the 'Institutional Model Index' the FRONTLOAD expression has three parameters e.g. (100, 30, 5). The first of these is the number of largest exams that are to be considered. Largest exams are specified by class size. If there are ties by size then exams occurring first in the data file are chosen. The second parameter is the number of last periods to take into account and which should be ideally avoided by the large exams. The third parameter is the penalty or weighting that should be added each time the soft constraint is violated. This allows the institution to attempt to ensure that larger exams occur earlier in the examination session. This constraint is very popular in practice as exams with more students enrolled take longer to mark. (See Sect. 4.9.5.)

3.3.6 Soft room penalty

Organisations often want to keep usage of certain rooms to a minimum. As with the 'Mixed Durations' component of the overall penalty, this part of the overall penalty should be calculated on a period by period basis. For each period, if a room used within the solution has an associated penalty, then the penalty for that room for that period is calculated by multiplying the associated penalty by the number of exams using that room. (See Sect. 4.9.6.)

3.3.7 Soft period penalty

Organisations often want to keep usage of certain time periods to a minimum. As with the 'Mixed Durations' and the 'Room Penalty' components of the overall penalty, this part of the overall penalty should be calculated on a period by period basis. For each period, the penalty is calculated by multiplying the associated penalty by the number of exams timetabled within that period. (See Sect. 4.9.7.)

These constraints are relatively complex, and certainly more complicated than those in previous models. To reduce potential ambiguities due to the above natural language description, in the next section, we provide mathematical programming definitions.

4 Integer programming formulation

The mathematical formulation given here is intended to provide a comprehensive and rigorous definition of the new model.[5] Accordingly, it was designed for compactness and (relative) clarity, and the encodings are not optimised towards giving the best results when presented to an integer programming solver. However, they are still usable, albeit only on

[5]The encoding (in OPL/CPLEX format) is available via http://www.cs.nott.ac.uk/~ajp/timetabling/exam/.

instances somewhat smaller than those provided as challenges in the competition (Burke et al. 2008; Parkes and Özcan 2010). They might also provide the basis for future improved encodings.

The problem description captures real-world constraints, and so is inherently rather long and complicated. This inevitably has the effect that ambiguities and misunderstandings can easily creep into implementations. Accordingly, for the competition, we implemented three separate solution validators and cross-checked their results. Two of these validators were coded directly and independently (and one used on the competition website), the third was based on encoding the formulation given in this section into Ilog's OPL/CPLEX.[6] It is then straightforward to create a solution validator by using a given solution to fix the primary decision variables specifying the period and room assignments. In this case, integer programming solvers easily (i.e. without branch-and-bound, but just preprocessing and the linear relaxation) determine whether the solution is feasible, and simultaneously produce the values for the various penalties. We extensively cross-checked the results of such direct implementations with the integer programming formulation. In any model as complicated as the one presented here, the chances of ambiguity, misunderstanding, or simple human-error increase dramatically, and so such extensive testing is tedious but necessary.

For the IP model, for simplicity of presentation, we will also treat the RELAXABLE-HARD constraints as normal hard constraints. This is reasonable as the competition instances happened to be feasible, and so it was possible to satisfy both STRICT-HARD and RELAXABLE-HARD constraints. It would be straightforward to convert the 'RELAXABLE-HARD' constraints to highly-penalised soft constraints, e.g. by the introduction of appropriate slack variables.

To help render the following formulation more readable we will follow the following conventions:

- Sets (of exams, etc.) are upper case.
- Parameters (quantities whose value is known or easily derivable from the input files) are lower case.
- Variables (quantities whose value is to be determined by the search) are upper case.
- When quantities such as size are associated with two different types, such as exam size and room size, then, rather than increase the usual plethora of symbols, we will indicate the "type" with a superscript. For example, s^E and s^R are used for exam and room sizes respectively.

4.1 Sets and parameters

The following sets and parameters are either directly given in the input file, or are straightforward to derive from it.

Exam related:

E: set of exams.
s_i^E: size of exam $i \in E$.
d_i^E: duration of exam $i \in E$.
f_i^E: a boolean that is 1 if exam i is subject to the front-load penalties, 0 otherwise.
D: the set of durations used $\bigcup_i d_i^E$.

[6] http://www.ilog.fr. (Now part of IBM.)

u_{id}^D: a boolean that is 1 if and only if exam i has "duration type" $d \in D$. We call them "types" because the duration values don't matter, only whether they are equal.

In the competition input file format, the "FRONTLOAD" entry specifies 3 parameters. The first parameter is the number of largest exams with the largest exams being specified by class size. This is used to select which exams are subject to the front load penalty, that is, the exams, i, for which $f_i^E = 1$. To be precise, the exams should be sorted by largest-first, with a secondary sort by earliest-index-first in the case of equal sized exams. The specified number of exams are then taken from the front of this sorted list and given $f_i^E = 1$, the remaining exams (if any) are given $f_i^E = 0$. (The other two parameters are used later.)

The duration type, u_{id}^D, is used for the no-mixed-duration penalty. For example, suppose all exams might have durations of either 120 or 180 minutes, then there would be two duration types, and we could have $d \in \{0, 1\}$ with $u_{i0}^D = 1$ if and only if exam i has duration 120, $u_{i1}^D = 1$ if and only if exam i has duration 180.

Students:

S: set of students.

Student enrollments are encoded by:

t_{is}: 1 if student s takes (is enrolled in) exam i, 0 otherwise.

Room related:

R: set of rooms.
s_r^R: size of room $r \in R$.
w_r^R: a weight that specifies the penalty for using room r.

Period related:

P: set of periods.
d_p^P: duration of period $p \in P$.
f_p^P: a boolean that is 1 if period p is subject to the FRONTLOAD penalties, 0 otherwise. The second parameter of the FRONTLOAD line in the input file is used to fix this. Starting with the latest period, the required number of periods are given $f_p^P = 1$.
w_p^P: a weight that specifies the penalty for using period p.
y_{pq}: a boolean that is 1 if periods p and q are in the same day, 0 otherwise.

4.2 Period-related hard constraints

The AFTER, EXAM_COINCIDENCE and EXCLUSION constraints use

H^{aft}: a set of pairs of exams.
H^{coin}: a set of pairs of exams.
H^{excl}: a set of pairs of exams.

For every pair $(e_1, e_2) \in H^{aft}$ exam e_1 must occur strictly after exam e_2. For every pair $(e_1, e_2) \in H^{coin}$ exams e_1 and e_2 must occur in the same period (though not necessarily the same room). For every pair $(e_1, e_2) \in H^{excl}$ exams e_1 and e_2 must *not* occur in the same period.

4.3 Room-related hard constraints

The EXCLUSIVE constraint uses

H^{sole}: a set of exams.

For every exam $e \in H^{sole}$, if exam e is assigned to period p and room r then e must be the sole occupier, i.e. no other exam can be assigned to both p and r. (Unless specified by an EXCLUSIVE rule, then, as standard in exam timetabling, exams are allowed to share rooms.)

4.4 Institutional weights and parameters

w^{2R}: weight for "two in a row".
w^{2D}: weight for "two in a day".
w^{PS}: weight for period spread (defaults to one as not currently specified in the input format, but included here for completeness).
w^{NMD}: weight for "no mixed duration".
w^{FL}: weight for the front load penalty.

The PERIODSPREAD line of the input format itself just specifies the parameter:

g: the period spread, the preferred minimal "gap" between exams for a student.

4.5 Variables

4.5.1 Primary decision variables

The binary (boolean) decision variables that fix the assignment are simply

$$X^P_{ip} = 1 \quad \text{if exam } i \text{ is in period } p, 0 \text{ otherwise} \tag{2}$$

$$X^R_{ir} = 1 \quad \text{if exam } i \text{ is in room } r, 0 \text{ otherwise} \tag{3}$$

4.5.2 Secondary variables

By secondary variables we mean those whose values will be directly forced given any legal assignment to primary variables. They are used to write the constraints and to compute the objective function.

In order to encode the room capacity constraints we will use the variable

$$X^{PR}_{ipr} = 1 \quad \text{if exam } i \text{ is in period } p \text{ and room } r, 0 \text{ otherwise} \tag{4}$$

and the intended meaning is that

$$\forall i \in E,\ \forall p \in P,\ \forall r \in R \quad X^{PR}_{ipr} \equiv X^P_{ip} X^R_{ir} \tag{5}$$

However, being non-linear this will not be enforced directly, but will just be forced by other (linear) constraints, specifically (15) and (16).

The penalties for violations of the various soft constraints are encoded as non-negative variables as follows

C^{2R}_s = "Two in a row" penalty for student s

C_s^{2D} = "Two in a day" penalty for student s
C_s^{PS} = "Period spread" penalty for student s
C^{NMD} = "No mixed duration" penalty
C^{FL} = "Front-load" penalty
C^{P} = "Soft period" penalty
C^{R} = "Soft room" penalty

These variables are all secondary in the sense that their values are forced by the primary decision variables and the constraints. (It follows that many need not be forced to be integer but can be relaxed to be real values if desired.)

We also remark that many of these variable are not strictly necessary, as the terms they will represent could be included directly into the objective. However, we keep them separate for the purposes of clarity, and because their values should correspond to values given by the validator on the competition website.

We also use the following variable:

$$U_{dpr}^{D} = 1 \quad \text{if duration type } d \text{ is used in period } p \text{ and room } r, \text{ 0 otherwise} \tag{6}$$

4.6 Objective and constraints

4.6.1 Objective

Minimise

$$\sum_{s \in S} \left(w^{2R} C_s^{2R} + w^{2D} C_s^{2D} + w^{PS} C_s^{PS} \right) + w^{NMD} C^{NMD} + w^{FL} C^{FL} + C^{P} + C^{R} \tag{7}$$

Notice that there are no separate weights for the room and period penalties C^R and C^P as the associated weights were already included in their definitions. Of course, the problem is inherently multi-objective (Burke et al. 2008), but this weighted sum approach is used here for simplicity. We also define the penalties for the entire set of students:

$$C^{2R} = \sum_{s \in S} C_s^{2R} \tag{8}$$

$$C^{2D} = \sum_{s \in S} C_s^{2D} \tag{9}$$

$$C^{PS} = \sum_{s \in S} C_s^{PS} \tag{10}$$

Minimisation is subject to the following hard constraints defining feasibility, and also to the constraints defining the soft penalties.

4.7 The STRICT-HARD constraints

Every exam is allocated to at least one room, and to at least one period:

$$\forall i \in E \quad \sum_{r \in R} X_{ir}^{R} \geq 1 \tag{11}$$

$$\forall i \in E \quad \sum_{p \in P} X_{ip}^{P} \geq 1 \tag{12}$$

Every exam is allocated to at most one room (exams cannot be "split"):

$$\forall i \in E \quad \sum_{r \in R} X_{ir}^{R} \leq 1 \tag{13}$$

Every exam is allocated to at most one period:

$$\forall i \in E \quad \sum_{p \in P} X_{ip}^{P} \leq 1 \tag{14}$$

To link the various X decision variables we enforce

$$\forall i \in E \; \forall p \in P \quad X_{ip}^{P} = \sum_{r \in R} X_{ipr}^{PR} \tag{15}$$

$$\forall i \in E \; \forall r \in R \quad X_{ir}^{R} = \sum_{p \in P} X_{ipr}^{PR} \tag{16}$$

4.8 The RELAXABLE-HARD constraints

Room capacities are always respected:

$$\forall p \in P \; \forall r \in R \quad \sum_{i \in E} s_i^E X_{ipr}^{PR} \leq s_r^R \tag{17}$$

Period durations are respected:

$$\forall p \in P \; \forall i \in E \quad d_i^E X_{ip}^{P} \leq d_p^P \tag{18}$$

In any period, any student is taking at most one exam:

$$\forall p \in P \; \forall s \in S \quad \sum_{i \in E} t_{is} X_{ip}^{P} \leq 1 \tag{19}$$

This enforces the usual conflict matrix between exams. The hard period constraints are enforced by:

$$\forall (i, j) \in H^{aft} \; \forall p, q \in P, \text{ with } p \leq q$$
$$X_{ip}^{P} + X_{jq}^{P} \leq 1 \tag{20}$$
$$\forall (i, j) \in H^{coin} \; \forall p \in P,$$
$$X_{ip}^{P} = X_{jp}^{P} \tag{21}$$
$$\forall (i, j) \in H^{excl} \; \forall p \in P,$$
$$X_{ip}^{P} + X_{jp}^{P} \leq 1 \tag{22}$$

The hard room constraints are enforced by

$$\forall i \in H^{sole} \; \forall j \in E, j \neq i \; \forall p \in P, \; \forall r \in R,$$
$$X_{ip}^{P} + X_{ir}^{R} + X_{jp}^{P} + X_{jr}^{R} \leq 3 \tag{23}$$

4.9 Soft constraints

Finally, we cover the penalty terms in the objective corresponding to violations of the soft constraints. We will use the term "pattern penalties" for those terms arising from restrictions on sequences of enrolment for each student; the two-in-a-row C^{2R}, two-in-a-day C^{2D}, and period-spread C^{PS} penalties. These pattern penalties are particularly awkward to encode and susceptible to misunderstanding: Hence, to enhance clarity we will first present them as non-linear constraints, and then give an equivalent linear encoding. (Note a non-linear version is actually reasonable to use as a solution validator because the non-linear terms are fixed to constant values when given a solution, and such a version can have the advantage of greater clarity and using fewer variables.)

4.9.1 Two in a row

If a student s is enrolled into two distinct exams i and j, and j occurs on the same day in the period immediately after the period used for i, then the penalty C_s^{2R} receives an increment of 1. Hence,

$$C_s^{2R} = \sum_{\substack{i,j \in E \\ j \neq i}} \sum_{\substack{p,q \in P \\ q=p+1 \,\&\, y_{pq}=1}} t_{is} t_{js} X_{ip}^P X_{jq}^P \tag{24}$$

Notice that there is no double counting. The condition on the periods is $q = p + 1$ rather than $|p - q| = 1$ and so the condition $j \neq i$ is needed rather than $j > i$. That is, we separately capture the cases 'i is before j' and 'i is after j'.

To convert to linear form observe that such a term as $X_{ip}^P X_{jq}^P$ is only relevant when the two exams i and j have students in common and so we only need consider conflicting pairs of exams. This suggests using edges of the associated (student-generated) conflict graph $G_C = (E, A)$ with vertices being the set E of exams. The set A of undirected edges, contains an edge (i, j) if and only if exams i and j conflict. Without loss of generality, and to prevent double counting, we assume $i < j$ for all $a = (i, j) \in A$. Define a weight, w_a^C, for each edge in A, to be the number of students taking both of the two exams:

$$\forall a = (i, j) \in A \quad w_a^C = \sum_{s \in S} t_{is} t_{js} \tag{25}$$

Introduce a new integer variable to encode the total penalty at each edge of the conflict graph:

$$C_a^{2R} = \begin{cases} 1 & \text{if edge } a = (i, j) \text{ for exams } i \text{ and } j \text{ incurs a two-in-a-row penalty} \\ 0 & \text{otherwise} \end{cases} \tag{26}$$

Then the overall penalty is the sum of such occurrences weighted by the number of students:

$$C^{2R} = \sum_{a \in A} w_a^C C_a^{2R} \tag{27}$$

The minimisation of C^{2R} within the overall objective will automatically force C_a^{2R} towards zero. Hence, it only remains to force it to be one when necessary, using the following *linear* constraints:

$$\begin{aligned} &\forall a = (i, j) \in A \; \forall p, q \in P \; |q - p| = 1 \,\&\, y_{pq} = 1 \\ &\quad X_{ip}^P + X_{jq}^P \leq 1 + C_a^{2R} \end{aligned} \tag{28}$$

Equations (27) and (28) can then replace (24).

4.9.2 Two in a day

If a student s is enrolled into two distinct exams i and j and these exams occur in non-consecutive periods on the same day, then the penalty C_s^{2R} receives an increment of 1. Hence,

$$C_s^{2D} = \sum_{\substack{i,j \in E \\ j \neq i}} \sum_{\substack{p,q \in P \\ q > p+1 \,\&\, y_{pq}=1}} t_{is} t_{js} X_{ip}^P X_{jq}^P \tag{29}$$

This is the same as for two-in-a-row except the $q = p + 1$ condition changed to $q > p + 1$. To convert to linear form, introduce new variables

$$C_a^{2D} = \begin{cases} 1 & \text{if edge } a = (i, j) \text{ for exams } i \text{ and } j \text{ incurs a two-in-a-day penalty} \\ 0 & \text{otherwise} \end{cases} \tag{30}$$

and enforce the constraints

$$C^{2D} = \sum_{a \in A} w_a^C C_a^{2D} \tag{31}$$

$$\begin{aligned} &\forall a = (i, j) \in A \; \forall p, q \in P \; |q - p| \geq 2 \,\&\, y_{pq} = 1 \\ &X_{ip}^P + X_{jq}^P \leq 1 + C_a^{2D} \end{aligned} \tag{32}$$

4.9.3 Period spread

If a student s is enrolled into two distinct exams i and j and these exams occur in distinct periods such that j is after i but is within the gap g, then the penalty C_s^{PS} receives an increment of 1. Again, double counting is prevented by putting a time order on the exams that contribute. Hence,

$$C_s^{PS} = \sum_{\substack{i,j \in E \\ j \neq i}} \sum_{\substack{p,q \in P \\ p < q \leq p+g}} t_{is} t_{js} X_{ip}^P X_{jq}^P \tag{33}$$

This is the same as for two-in-a-row or day except that the conditions on the two periods p and q changed to $q \in \{(p + 1), \ldots, (p + g)\}$. To convert to linear form, introduce new variables

$$C_a^{PS} = \begin{cases} 1 & \text{if edge } a = (i, j) \text{ between exams } i \text{ and } j \text{ incurs a period spread penalty} \\ 0 & \text{otherwise} \end{cases} \tag{34}$$

and enforce the constraints:

$$C^{PS} = \sum_{a \in A} w_a^C C_a^{PS} \tag{35}$$

$$\begin{aligned} &\forall a = (i, j) \in A \; \forall p, q \in P \; 1 \leq |q - p| \leq g \\ &X_{ip}^P + X_{jq}^P \leq 1 + C_a^{PS} \end{aligned} \tag{36}$$

4.9.4 No mixed durations

We need to force U^D_{dpr} to be non-zero whenever some exam with duration type d uses period p and room r:

$$\forall d \in D\ \forall i \in E \text{ with } u^D_{id} = 1\ \forall p \in P\ \forall r \in R$$
$$U^D_{dpr} \geq X^P_{ip} + X^R_{ir} - 1 \tag{37}$$

The period-room pair pr receives a non-negative penalty, C^{NMD}_{pr}, which is the maximum of zero and the excess above one of the total number of duration types assigned to it:

$$\forall p \in P\ \forall r \in R$$
$$1 + C^{NMD}_{pr} \geq \sum_{d \in D} U^D_{dpr} \tag{38}$$
$$C^{NMD}_{pr} \geq 0 \tag{39}$$

The overall penalty is

$$C^{NMD} = \sum_{p \in P} \sum_{r \in R} C^{NMD}_{pr} \tag{40}$$

The minimisation in the overall objective will force U^D_{dpr} and C^{NMD}_{pr} to be the intended minimal values consistent with the assignment.

4.9.5 Front load

The front load penalty applies to exams i with $f^E_i = 1$ and assigned to periods with $f^P_p = 1$, and so

$$C^{FL} = \sum_{i \in E} \sum_{p \in P} f^E_i f^P_p X^P_{ip} \tag{41}$$

4.9.6 Soft period penalties

These are simply enforced by:

$$C^P = \sum_{p \in P} \sum_{i \in E} w^P_p X^P_{ip} \tag{42}$$

4.9.7 Soft room penalties

Finally, the soft room penalties are simply enforced by:

$$C^R = \sum_{r \in R} \sum_{i \in E} w^R_r X^R_{ir} \tag{43}$$

5 Competition data, solvers and results

In this section we briefly describe the instances that were used, the solvers submitted, and their final ranking in the competition.

Table 1 Basic properties of the public instances of the examination timetabling track. Density is the conflict density given as a percentage. HC refers to the numbers of Hard Constraints

Instance	Density	Exams	Students	Periods	Rooms	Period HC	Room HC
Exam_1	5.05	607	7891	54	7	12	0
Exam_2	1.17	870	12743	40	49	12	2
Exam_3	2.62	934	16439	36	48	170	15
Exam_4	15.0	273	5045	21	1	40	0
Exam_5	0.87	1018	9253	42	3	27	0
Exam_6	6.16	242	7909	16	8	23	0
Exam_7	1.93	1096	14676	80	15	28	0
Exam_8	4.55	598	7718	80	8	20	1

5.1 The datasets

All datasets used as part of the competition were taken from a variety of Universities in Europe, Australasia and America. Countries covered include UK, Portugal, Australia, and the USA, and so we think are representative of a large range of different systems. The data (and institution names) were anonymised for the purpose of data protection and competition use. Four datasets were available from the start, four were released two weeks prior to the end of the competition, and the final four were kept as 'hidden' datasets for internal testing and ranking purposes in line with the competition rules. Table 1 gives some basic properties, such as size, of the eight public instances.

Recall from Sect. 2.2 that any solution violating one or more hard or required constraints is called infeasible. However, It turned out that for this data set they all had feasible solutions, and furthermore most solvers usually found a feasible solution. This is perhaps not surprising as they were real-world problems and so, for example, resources would have been selected so that feasible solutions exist. Hence, although, 'post facto' for the data set used, the distance to feasibility was not a direct influence on the results; at the time of solver submission, submitters would not know that the hidden instances were necessarily all feasible, and so DTF might have been vital for them to reduce. Furthermore, from the competition website, and in reference to the framework of Sect. 2.2:

> Although, the nature of the practical problem described here usually leads to feasibility being found quite easily, this is not necessarily always the case in practice. It was felt essential that this information was included here to allow solution evaluation to be consistent across all tracks of the competition and in order to establish an evaluation method that can be built upon for the future.

That is, the general framework was also retained as we believe it likely to be useful in future and future datasets might well require some hard constraints to be violated.

5.2 The winning solvers

The top five submissions, in order of their final placement, and with a brief description of the methods used were:

1. Tomas Müller (USA). This is a constraint based system using forward checking and also an improvement phase (Müller 2009).

2. Christos Gogos (Greece). This used GRASP together with local improvements (Gogos et al. 2010).
3. Mitsunori Atsuta, Koji Nonobe, and Toshihide Ibaraki (Japan). This used a constraint satisfaction solver together with tabu search and iterated local search (Atsuta et al. 2008).
4. Geoffrey De Smet (Belgium). This used a local search based solver (Smet 2008).
5. Nelishia Pillay (South Africa). This used a 'developmental approach' that mimicked the processes of cell biology (Pillay 2008).

Fuller descriptions of the submissions, and also details of the scores obtained are available from the ITC2007 website and McCollum et al. (2010). It is noteworthy that the winning submission of Müller was also submitted to the other two tracks. It also won the other track, on pre-enrollment curriculum timetabling, that used real-data, though did not win the track on post-enrollment course timetabling that used artificial data. This does tend to support the case that real and artificial data can lead to different solution methodologies, and so emphasises the importance of using real data.

6 Potential extensions of the model

Although the model presented in this paper is much richer than previous models, it does not necessarily capture everything that a user might want. Hence here we briefly discuss a few potentially useful extensions.

The first, and probably the easiest, extension would be simply to allow multiple copies of the FRONTLOAD and PERIODSPREAD declarations within the Institutional model. Each one would be computed separately and the associated penalties summed together. Since it just uses existing constraints then it would probably be relatively easy to implement in existing solvers. Multiple copies of FRONTLOAD would allow a finer control over the placement of the exams by size. For example, there could be a strong penalty on the very largest exams not being front loaded, but also a milder penalty might be used to keep medium sized exams away from the very end of the exam session.

Multiple copies of PERIODSPREAD would also allow the user a finer control over the spreading of the exams. However, from the research perspective it is particularly interesting in that it would allow the reproduction of the spread penalties, or proximity costs (Carter et al. 1996) commonly used in the literature with the Toronto data. Specifically, the standard penalty is 2^{5-g} for a gap of g, giving a sequence $\{16, 8, 4, 2, 1\}$ and this is equivalent to

```
PERIODSPREAD 1 8
PERIODSPREAD 2 4
PERIODSPREAD 3 2
PERIODSPREAD 4 1
PERIODSPREAD 5 1
```

The first and second numbers are the gap and penalty. Hence for example a gap of 3 would get a total penalty of $2 + 1 + 1 = 4$ matching the standard 2^{5-3}.

At present the splitting of exams between rooms is not permitted, and a potential modification would be to allow it with some specified (large) penalty for each split. However, this is probably not the highest priority as the current format does allow some splitting to be managed by creating multiple sessions. The different sessions of the same exam could be forced to be at the same time using an EXAM_COINCIDENCE constraint, but this would still allow them to be in different rooms. (This would be a reasonable solution when some students have special room requirements due to disabilities, etc.)

Finally, users might want to be able to specify that some exam (or exams) should preferably be in some room (or rooms). Specific penalties for each potential exam-room assignment would be a more drastic extension. However, it is perhaps worth noting that such location based penalties are less likely to be important than they would be for classes. People attend classes regularly and so minimising travel is likely to be much more important than for the relatively rare attendance at an exam.

7 Conclusion and discussion

We have presented a formal model of the examination timetabling problem as it appears in many institutions. As used in the examination track of the 2nd International Timetabling Competition, the formulation introduces many aspects of practical real world implementations. This work therefore serves to provide a firm basis for setting the research agenda for continual development in the field. In doing so the authors believe the formulation significantly addresses the gap which exists between the efforts made in research and the requirements of institutions.

An integer programming encoding was given in order to formally define the problem. Although not capable of solving the competition instances, it has recently been used in order to solve smaller instances. In Burke et al. (2008) it was used to explore the multi-objective aspects of a subset of a real problem. In Parkes and Özcan (2010) instances from Yeditepe University in Turkey were encoded (giving more evidence for the general utility of the framework) and the IP model used to exactly solve some of the smaller ones. Of course, it would be good to improve the encoding and also to consider other IP methods such as branch and cut.

In providing this formulation, it is pointed out that minimising the number of periods is not considered to be an objective, because in the authors' experience, educational institutions manage the process by using set times for the examination session. That is not to say, of course, that this is not a major issue in relation to planning examination sessions. Also, we have delayed until future work, the full investigation and explanation of which constraints are absolute, and which can be broken to give a "distance to feasibility". Such an investigation will allow the formulation provided here to cover not just the typical institutions and instances but also less common instances of the problem. For example, when feasibility is not possible for a given problem, institutions make decisions relating to either the structure of the time periods, the availability of the resources or indeed the events to be timetabled. It is envisaged that issues relating to this issue of "repairing infeasibility" will also be analysed in future work.

We emphasize that definition of these soft constraints is based on real-world experience, and their introduction and precise formulation for the academic community provides the core contribution of this paper.

References

Ahmadi, S., Barone, R., Cheng, P., Cowling, P., & McCollum, B. (2003). Perturbation based variable neighbourhood search in heuristic space for examination timetabling problem. In *Proceedings of multidisciplinary international scheduling: theory and applications (MISTA 2003)* (pp. 155–171). Nottingham, August 13–16. ISBN: 0-9545821-2-8.

Al-Yakoob, S., Sherali, H., & Al-Jazzaf, M. (2010). A mixed-integer mathematical modeling approach to exam timetabling. *Computational Management Science*, *7*, 19–46. doi:10.1007/s10287-007-0066-8.

Asmuni, H., Burke, E. K., Garibaldi, J. M., & McCollum, B. (2006). A novel fuzzy approach to evaluate the quality of examination timetabling. In *Proceedings of the 6th international conference on the practice and theory of automated timetabling (PATAT 2006)* (pp. 82–102). Brno, Czech Republic, August 30th–September 1st 2006.

Atsuta, M., Nonobe, K., & Ibaraki, T. (2008). *ITC-2007 track2: an approach using general CSP solver*. Available from the ITC-2007 website. http://www.cs.qub.ac.uk/itc2007/winner/bestcoursesolutions/Atsuta_et_al.pdf.

Azimi, Z. N. (2005). Hybrid heuristics for examination timetabling problem. *Applied Mathematics and Computation*, *163*(2), 705–733.

Borning, A., Duisberg, R., Freeman-Benson, B., Kramer, A., & Woolf, M. (1987). Constraint hierarchies. In N. Meyrowitz (Ed.), *Proceedings of the conference on object-oriented programming systems, languages, and applications (OOPSLA)* (Vol. 22, pp. 48–60). New York: ACM.

Borning, A., Freeman-Benson, B., & Wilson, M. (1992). Constraint hierarchies. *LISP and Symbolic Computation*, *5*, 223–270.

Burke, E. K., & Newall, J. (2004). Solving examination timetabling problems through adaptation of heuristic orderings. *Annals of Operations Research*, *129*, 107–134.

Burke, E. K., Newall, J., & Weare, R. F. (1996). A memetic algorithm for university exam timetabling. In E. K. Burke & P. Ross (Eds.), *Lecture notes in computer science: Vol. 1153. The practice and theory of automated timetabling* (pp. 241–250). Berlin: Springer.

Burke, E. K., Petrovic, S., & Qu, R. (2002). Case-based heuristic selection for examination timetabling. In *Proceedings of the 4th Asia-Pacific conference on simulated evolution and learning (SEAL 2002)* (pp. 277–281). Singapore, Nov 18–22 2002.

Burke, E. K., Hart, E., Kendall, G., Newall, J., Ross, P., & Schulenburg, S. (2003). Hyper-heuristics: an emerging direction in modern search technology. In *Handbook of meta-heuristics* (pp. 457–474). Dordrecht: Kluwer Academic.

Burke, E. K., Bykov, Y., Newall, J., & Petrovic, S. (2004). A time-predefined local search approach to exam timetabling problems. *IIE Transactions*, *36*, 509–528.

Burke, E. K., Dror, M., Petrovic, S., & Qu, R. (2005). Hybrid graph heuristics within a hyper-heuristic approach to exam timetabling problems. In *The next wave in computing, optimization, and decision technologies* (pp. 79–91). Berlin: Springer.

Burke, E. K., Petrovic, S., & Qu, R. (2006). Case based heuristic selection for timetabling problems. *Journal of Scheduling*, *9*(2), 115–132.

Burke, E. K., McCollum, B., Meisels, A., Petrovic, S., & Qu, R. (2007). A graph-based hyper-heuristic for educational timetabling problems. *European Journal of Operational Research*, *176*, 177–192.

Burke, E. K., McCollum, B., McMullan, P., & Parkes, A. J. (2008). Multi-objective aspects of the examination timetabling competition track. In *Proceedings of PATAT 2008*. Montreal, Canada, August 2008.

Burke, E. K., Eckersley, A. J., McCollum, B., Petrovic, S., & Qu, R. (2010). Hybrid variable neighbourhood approaches to university exam timetabling. *European Journal of Operational Research*, *206*, 46–53.

Callennec, B. L., & Boulic, R. (2004). Interactive motion deformation with prioritized constraints. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on computer animation, SCA'04* (pp. 163–171). Aire-la-Ville, Switzerland, Switzerland. Aire-la-Ville: Eurographics Association.

Carter, M. W., Laporte, G., & Lee, S. Y. (1996). Examination timetabling: algorithmic strategies and applications. *The Journal of the Operational Research Society*, *47*(3), 373–383.

David, P. (1998). A constraint-based approach for examination timetabling using local repair techniques. In E. K. Burke & M. W. Carter (Eds.), *Springer lecture notes in computer science: Vol. 1408. Practice and theory of automated timetabling: selected papers from the 2nd international conference* (pp. 169–186).

de Werra, D., Asratian, A. S., & Durand, S. (2002). Complexity of some special types of timetabling problems. *Journal of Scheduling*, *5*, 171–183.

Di Gaspero, L. (2002). Recolour, shake and kick: a recipe for the examination timetabling problem. In E. K. Burke & P. D. Causmaecker (Eds.), *Proceedings of the 4th international conference on the practice and theory of automated timetabling*. KaHo St.-Lieven, Gent, Belgium (pp. 404–407).

Di Gaspero, L., & Schaerf, A. (2001). Tabu search techniques for examination timetabling. In E. K. Burke & W. Erben (Eds.), *Lecture notes in computer science (LNCS): Vol. 2079. Practice and theory of automated timetabling: selected papers from the 3rd international conference* (pp. 104–117).

Di Gaspero, L., McCollum, B., & Schaerf, A. (2007). *The second international timetabling competition (ITC-2007): curriculum-based course timetabling (track 3)* (Tech. Rep. QUB/IEEE/Tech/ITC2007/CurriculumCTT/v1.0/1) August 2007, Queen's University Belfast. http://www.cs.qub.ac.uk/itc2007/.

Dowsland, K. A., & Thompson, J. (2005). Ant colony optimization for the examination scheduling problem. *The Journal of the Operational Research Society*, *56*(4), 426–438.

Dubois, D., Fargier, H., & Prade, H. (1996). Possibility theory in constraint satisfaction problems: handling priority, preference and uncertainty. *Applied Intelligence*, *6*, 287–309.

Duong, T. A., & Lam, K. H. (2004). Combining constraint programming and simulated annealing on university exam timetabling. In *Proceedings of the 2nd international conference in computer sciences, research, innovation & vision for the future (RIVF2004)* (pp. 205–210). Hanoi, Vietnam, February 2004.

Gogos, C., Alefragis, P., & Housos, E. (2010). An improved multi-staged algorithmic process for the solution of the examination timetabling problem. *Annals of Operations Research*. doi:10.1007/s10479-010-0712-3

Henz, M., Yap, R. H. C., Lim, Y. F., Lua, S. C., Walser, J. P., & Shi, X. P. (2004). Solving hierarchical constraints over finite domains with local search. *Annals of Mathematics and Artificial Intelligence*, *40*, 283–301.

Lewis, R., Paechter, B., & McCollum, B. (2007). *Post enrolment based course timetabling: a description of the problem model used for track two of the second international timetabling competition* (Cardiff Working Papers in Accounting and Finance A2007-3), Cardiff Business School, Cardiff University, August 2007.

McCollum, B. (2007). A perspective on bridging the gap between theory and practice in university timetabling. In *Lecture notes in computer science (LNCS): Vol. 3867. Revised selected papers of PATAT 2006, Proceedings of the 6th international conference on the practice and theory of automated timetabling* (pp. 3–23).

McCollum, B., McMullan, P., Burke, E. K., Parkes, A. J., & Qu, R. (2007). *The second international timetabling competition: examination timetabling track* (Tech. Rep. QUB/IEEE/Tech/ITC2007/Exam/v4.0/17.2007), Queen's University Belfast. http://www.cs.qub.ac.uk/itc2007/.

McCollum, B., Schaerf, A., Paechter, B., McMullan, P., Lewis, R., Parkes, A. J., Gaspero, L. D., Qu, R., & Burke, E. K. (2010). Setting the research agenda in automated timetabling: the second international timetabling competition. *INFORMS Journal on Computing*, *22*(1), 120–130.

Merlot, L. T. G., Boland, N., Hughes, B. D., & Stuckey, P. J. (2003). A hybrid algorithm for the examination timetabling problem. In *Springer lecture notes in computer science: Vol. 2740. Practice and theory of automated timetabling: selected papers from the 4th international conference* (pp. 207–231).

Meseguer, P., Bouhmala, N., Bouzoubaa, T., Irgens, M., & Sánchez, M. (2003). Current approaches for solving over-constrained problems. *Constraints*, *8*, 9–39. doi:10.1023/A:1021902812784.

Metaheuristics Network, International timetable competition 2002 (2003). http://www.idsia.ch/Files/ttcomp2002/. Organised by the metaheuristics network, http://www.metaheuristics.net/ and PATAT 2002 http://www.asap.cs.nott.ac.uk/patat/patat02/patat02.shtml. Accessed April 2008.

Müller, T. (2009). ITC2007 solver description: a hybrid approach. *Annals of Operations Research*, *172*, 429–446.

Mumford, C. L. (2007). An order based evolutionary approach to dual objective examination timetabling. In *Proceedings of the 2007 IEEE symposium on computational intelligence in scheduling (CI-Sched 2007)*, Honolulu, Hawaii, 1–5th April.

Paquete, L., & Stützle, T. (2002). Empirical analysis of tabu search for the lexicographic optimization of the examination timetabling problem. In E. Burke & P. D. Causmaecker (Eds.), *Proceedings of the 4th international conference on practice and theory of automated timetabling*.

Parkes, A. J., & Özcan, E. (2010). Properties of Yeditepe examination timetabling benchmark instances. In *Proc. of the 8th international conference on the practice and theory of automated timetabling (PATAT 2010)*.

Pillay, N. (2008). *A developmental approach to the examination timetabling problem*. http://www.cs.qub.ac.uk/itc2007/winner/bestexamsolutions/pillay.pdf.

Qu, R., Burke, E. K., McCollum, B., Merlot, L., & Lee, S. (2009). A survey of search methodologies and automated system development for examination timetabling. *Journal of Scheduling*, *12*(1), 55–89.

Ross, P., Corne, D., & Terashima-Marín, H. (1996). The phase transition niche for evolutionary algorithms in timetabling. In E. K. Burke & M. A. Trick (Eds.), *Lecture notes in computer science: Vol. 1153. Selected papers from the first international conference on the theory and practice of automated timetabling (PATAT 95)* (pp. 309–324). New York: Springer.

Ross, P., Marin-Blazquez, J., & Hart, E. (2004). Hyper-heuristics applied to class and exam timetabling problems. In *Proceedings of the 2004 congress on evolutionary computation (CEC2004)* (pp. 1691–1698).

Smet, G. D. (2008). *ITC 2007—examination track*. Available from the ITC-2007 website. http://www.cs.qub.ac.uk/itc2007/.

Thompson, J., & Dowsland, K. (1998). A robust simulated annealing based examination timetabling system. *Computers & Operations Research*, *25*, 637–648.

Tsang, E., Mills, P., & Williams, R. (1999). A computer aided constraint programming system. In *The 1st international conference on the practical application of constraint technologies and logic programming (PACLP)* (pp. 81–93).

van Loon, J. N. M. (1981). Irreducibly inconsistent systems of linear inequalities. *European Journal of Operational Research*, *8*(3), 283–288.

White, G. M., & Xie, B. S. (2001). Examination timetables and tabu search with longer-term memory. In E. K. Burke & W. Erben (Eds.), *Practice and theory of automated timetabling: selected papers from the 3rd international conference*.

Wong, T., Cote, P., & Gely, P. (2002). Final exam timetabling: a practical approach. In *IEEE Canadian conference on electrical and computer engineering (CCECE 2002)* (Vol. 2, pp. 726–731).