Integer Programming Methods for Large Scale Practical Classroom Assignment Problems

Antony Phillips[a,*], Hamish Waterer[b], Matthias Ehrgott[c], David Ryan[a]

[a]Department of Engineering Science, The University of Auckland, New Zealand
[b]Centre for Optimal Planning and Operations, The University of Newcastle, Australia
[c]Department of Management Science, Lancaster University, United Kingdom

Abstract

In this paper we present an integer programming method for solving the Classroom Assignment Problem in University Course Timetabling. We introduce a novel formulation of the problem which generalises existing models and maintains tractability even for large instances. The model is validated through computational results based on our experiences at the University of Auckland, and on instances from the 2007 International Timetabling Competition. We also expand upon existing results into the computational difficulty of room assignment problems.

Keywords: University course timetabling, classroom assignment, integer programming, lexicographic optimisation

1. Introduction

University course timetabling is a large resource allocation problem, in which both times and rooms are determined for each class meeting. Due to the difficulty and size of modern timetabling problems, much of the academic literature proposes purely heuristic solution methods. However, in recent years, integer programming (IP) methods have been the subject of increased attention. At the time of writing, MirHassani and Habibi (2013) have conducted the most recent survey into university course timetabling, which covers some of the IP approaches, as well as the most popular heuristic paradigms.

Some integer programming studies have been conducted in a practical setting (e.g. Schimmelpfeng and Helber, 2007; van den Broek et al., 2009), although inevitably the models are only solved for either a small university, or a single department at a larger university. These are unsuitable for large universities where the majority of teaching space is shared between departments and faculties.

While it is not yet possible to solve a large practical course timetabling problem to optimality (Burke et al., 2008), the problem can be decomposed into a timetable generation problem followed by a classroom assignment problem (also known as "times first, rooms second"). In our experience, this decomposition is commonly used in practice. Faculties

*Corresponding author
Email address: `antony.phillips@auckland.ac.nz` (Antony Phillips)

or departments may prefer to generate a timetable for their courses, and retain control over unique requirements and preferences. This is contrasted to the room assignment, which must be performed centrally in institutions with shared teaching space. For this reason, in some institutions the classroom assignment problem is the only part of course timetabling which uses computer-aided decision making.

The most elementary formulation of the classroom assignment problem attempts to find a feasible assignment for a set of classes (or *events*) to a set of rooms. A simple measure of quality may also be used, where a cost is assigned for all possible event-to-room assignments. This formulation allows each time period to be modelled as an independent assignment problem, which can be solved in polynomial time (Carter and Tovey, 1992). This is equivalent to finding a maximum weighted bipartite matching between the set of events and the set of rooms, as implemented by Lach and Lübbecke (2008).

The problem becomes more complex if the events vary in duration, and each event must occupy only one room for the entirety of this duration (referred to as *contiguous room stability*). Although this is a more practically useful problem, the interdependencies between blocks of contiguous time periods cause this problem to be NP-hard even for just two time periods (Carter and Tovey, 1992). Glassey and Mizrach (1986) propose an integer programming formulation for this problem, yet do not solve it due to the prohibitive number of variables (relative to available computational resources), and the possibility of non-integer solutions to the LP relaxation. Instead, they propose a simple heuristic procedure.

Gosselin and Truchon (1986) also approach the problem (with contiguous room stability) using an integer programming formulation, and aggregate the variables to reduce the problem size. When solving their model, they remark that the simplex method yielded integer solutions to the LP relaxation in every test case. Carter (1989) conducts the most advanced study into this problem, where the contiguous room stability requirement is enforced using an iterative Lagrangian relaxation method. A wide range of quality measures are considered which are weighted and combined (scalarised) into a single objective function. The author also outlines the experience of satisfying staff and administration requirements while implementing this method at the University of Waterloo, Canada.

The most complex formulations of the classroom assignment problem are able to address quality measures which cause interdependencies between any subset of time periods, rather than just a contiguous block. The most common example is minimising the number of different rooms used by each course (referred to as *course room stability*), which causes the problem to be NP-hard (Carter and Tovey, 1992). As part of a broader work, Qualizza and Serafini (2005) propose an integer programme to solve this problem, although they do not include results. Lach and Lübbecke (2012) also propose an integer programme which models course room stability, as part of their solution to the problems posed in Track 3 of the 2007 International Timetabling Competition, or ITC (Di Gaspero et al., 2007). Although Lach and Lübbecke (2012) include comprehensive computational results, they are only concerned with the abstract problems from the ITC, and only consider a single measure of quality. In practice it is often desirable to consider multiple measures of quality simultaneously.

We also acknowledge alternative definitions of the classroom assignment problem within the scope of university timetabling. Dammak et al. (2006) and Elloumi et al. (2014) use heuristic methods to address classroom assignment in the context of examination timetabling, where it is possible to assign more than one event to a room (in any given time period). Mirrazavi et al. (2003) apply integer goal programming to a similar problem where multiple 'subjects' are assigned together into rooms.

In this paper we propose a novel integer programming based method for the classroom assignment problem of university course timetabling. Our method is demonstrated to be versatile in terms of modelling power, capable of handling multiple competing quality measures, and tractable for large practical problems. We validate the method with computational results on data from the University of Auckland, New Zealand, and offer an insight into the timetabling process used until 2010. We also present computational results for the problems from the 2007 International Timetabling Competition (ITC). Through this work, we are able to expand upon previous results into the difficulty of classroom assignment problems. Although most variants of the classroom assignment problem found in practice are NP-hard, we demonstrate why many instances can be solved efficiently.

The remainder of this paper is organised as follows. Section 2 provides a simple example of a classroom assignment problem, outlines a general integer programming model, and introduces some common quality measures. Section 3 provides an insight into the matrix structure of the integer programme and demonstrates how fractions can arise in the linear programming relaxation. This allows us to identify which practical situations and quality measures will make the integer programme either easier or more difficult to solve. Section 4 details a timetabling system used at the University of Auckland and explains how practical considerations are modelled within our approach. In Section 5 we present the results of our method on data from the University of Auckland, and the ITC problems. We also address some shortcomings of the ITC problems which suggest they are not representative in size or structure of most practical timetabling problems. Finally, Section 6 outlines the main conclusions of our work, and future research directions.

2. A Set Packing Model for Classroom Assignment

In this section we introduce the classroom assignment problem using a small example, and demonstrate how this type of problem can be modelled as a maximum set packing problem (Nemhauser and Wolsey, 1988). To solve this problem, we propose an integer programming based approach, which provides a certainty of the feasibility (or infeasibility) of the room assignment and of the solution quality. Integer programming for set packing problems has also been applied to small instances of the broader course timetabling problem (Avella and Vasil'Ev, 2005).

To handle different measures for quality, our model is solved sequentially for a prescribed series of solution quality measures. The quality with respect to each measure is preserved in subsequent solutions using an explicit constraint. In the terminology of multiobjective optimisation (Ehrgott, 2005), this is a lexicographic optimisation algorithm, which is guaranteed to find a Pareto optimal solution i.e. no quality measure can be improved without reducing the quality of at least one other measure.

In practical timetabling, it may not always be possible to find a room for all teaching events (due to the structure of the timetable) i.e. the room assignment is infeasible. To handle this situation, our approach will find an efficient *partial* room assignment which makes the best possible use of the available rooms. It will also identify specifically which time periods are over-booked and which sizes (and types) of rooms are in shortage in each period. This information is important when timetablers decide how to modify the timetable, and the related analytics may also be of use to other administrative parties to understand the bottlenecks in the system.

2.1. Introductory Example Problem

A classroom assignment problem arises where a set of teaching events (e.g. lectures), each require the use of a suitable room in their prescribed time period. Each event is part of a course, which defines the size of the event (i.e. the course enrolment) and the room attributes which are required for this event. Table 1 contains this data on the courses and events for an example problem. Precise definitions for the terminology and notation used in column headers is provided in Section 2.2.

Course (c)	Size ($size_c$)	Room Attributes (att_c)	Course Events (e)	Time Period (T_e)
c_1	125	——	e_1	t_1
c_2	60	Demonstration Bench	e_1	t_1
			e_2	t_2
c_3	60	——	e_1	t_1
			e_2	t_2
			e_3	t_3
c_4	60	Demonstration Bench	e_1	t_2
			e_2	t_3

Table 1: Course and Event Data

Table 2 contains the data on which rooms are available. Each room has a size (i.e. the maximum student capacity), a set of room attributes, and a set of time periods when this room may be used.

Room (r)	Size ($size_r$)	Room Attributes (att_r)	Available Time Periods (T_r)
r_1	150	——	t_1, t_2, t_3
r_2	75	Demonstration Bench	t_1, t_2, t_3
r_3	75	Demonstration Bench	t_1, t_2, t_3

Table 2: Room Data

A simple model for the room assignment problem uses variables corresponding to a feasible event-to-room assignment. However, a more general approach models the assignment of a set of events, or pattern, to a feasible room.

Processing the data from Tables 1 and 2, we can generate the core problem data for Example 1 in Table 3. For each course, we show which time period each course event is held in, the feasible rooms for these events (determined by the room size and attributes), and the course patterns (all possible subsets of course events).

Example 1. A small classroom assignment problem

Course (c)	t_1	t_2	t_3	Feasible Rooms (R_c)	Course Patterns (P_c)
c_1	e_1			r_1	$\{e_1\}$
c_2	e_1	e_2		r_2, r_3	$\{e_1\}$, $\{e_2\}$, $\{e_1, e_2\}$
c_3	e_1	e_2	e_3	r_1, r_2, r_3	$\{e_1\}$, $\{e_2\}$, $\{e_3\}$, $\{e_1, e_2\}$, $\{e_1, e_3\}$, $\{e_2, e_3\}$, $\{e_1, e_2, e_3\}$
c_4		e_1	e_2	r_2, r_3	$\{e_1\}$, $\{e_2\}$, $\{e_1, e_2\}$

Table 3: Processed Problem Data

Course (c)	Pattern (p)	Room (r)
c_1	$\{e_1\}$	r_1
c_2	$\{e_1, e_2\}$	r_2
c_3	$\{e_1\}$	r_3
	$\{e_2\}$	r_1
	$\{e_3\}$	r_2
c_4	$\{e_1, e_2\}$	r_3

Table 4: A Feasible Solution

Table 4 gives a feasible solution to this problem, where patterns are assigned to feasible rooms, and the pattern-to-room assignments uphold that each room is used at most once in each time period. If our objective is to maximise the number of events assigned, this solution is clearly optimal, with all events assigned. If we also want to minimise the number of different rooms used by each course, the solution can be improved by assigning pattern $\{e_2, e_3\}$ of course c_3 to room r_1.

2.2. Notation

A teaching *event* e is a meeting between staff and students (e.g. a lecture), which requires a room for the duration of one time period in the timetabling domain (typically one week). Let E denote the set of events. A *course* c is a set of related events, which require a room of size at least $size_c$ (measured by the number of seats) and also possessing at least the room attributes att_c. Let C denote the set of all courses. The set of courses C partitions the set of events E, i.e. $E = \cup_{c \in C} c$, and $c_1 \cap c_2 = \emptyset$ for all $c_1, c_2 \in C$.

A meeting *pattern* p is defined to be a subset of events for a given course that will be assigned the same room. For course c, let P_c denote the set of all its patterns, the power set of course events. Let $length_c$ and $length_p$ denote the number of events in a course and pattern respectively. As a power set, P_c will feature $2^{length_c} - 1$ elements, which potentially could be large. However, in practice, the number of events per course is usually quite small (for example, averaging between 2 and 3 at the University of Auckland). Let P denote the set of all patterns, i.e. $P = \bigcup_{c \in C} P_c$. Note that while each pattern p uniquely identifies a set of events, an event is usually in more than one pattern. This is evident in Example 1, where Table 3 shows the events in each pattern for all courses. Let P_e denote the set of all patterns which contain event e, i.e. $P_e = \{p \in P \colon e \in p\}$.

Let R denote the set of *rooms* in the pool of common teaching space, where $size_r$ and att_r correspond to the room size and set of room attributes for a room r. R_c represents the set of rooms which are suitable for events of course c, i.e. $R_c = \{r \in R\colon size_r \geq size_c, att_r \supseteq att_c\}$. Using this definition, the course and room data from Tables 1 and 2 respectively can be processed to generate R_c for each course in Table 3. A pattern p of course c will have the set of feasible rooms for this pattern R_p, as a subset of R_c. For the rooms within R_c, a course's preference for a particular room is given by some preference function $Pref(c, r)$. This is usually used to place courses into buildings as close as possible to their teaching department, but may be used for any measure of preference e.g. more modern rooms.

Let A denote the set of all room attributes, i.e. $A = \bigcup_{r \in R} att_r$. In addition to physical room attributes, this set may contain abstract auxiliary attributes to assist with modelling. For example, a room may possess the attribute of being within a given maximum geographical distance from a particular teaching department. Abstract room attributes may also be used if a course wishes to avoid an undesirable room attribute. In the general case, this can be modelled by generating a complementary room attribute which corresponds to 'not-possessing' the undesirable attribute. The set of rooms is thus partitioned by those with the original undesirable attribute, and those with the complementary attribute. In many cases, partitions of the set of rooms already exist (e.g. if rooms are designated as one of several types), in which case requesting a room with one attribute automatically precludes being assigned a room with the other attributes.

Let T denote the set of all usable time *periods* in the timetabling domain, which are of a common duration (often one hour) and are non-overlapping. For practical problems we also introduce T_r to denote the set of time periods for which room r is available for teaching. Due to other prescheduled events, every room may have its unique set of available time periods. Each event $e \in E$ occurs during a prescribed time period T_e given by the timetable. For each pattern $p \in P$, let T_p denote the set of time periods this pattern features in, i.e. $T_p = \bigcup_{e \in p} T_e$.

Although many class meetings take place in a single period, some may be two or more periods long (e.g. tutorials or labs) which we refer to as *long* events. Long events require one event $e \in E$ for each time period they are held in. If a long event requires the same room for its entire duration, we refer to this requirement as *contiguous room stability*. This is enforced by pruning the set of patterns for this course, P_c, to only include patterns which contain all or none of these events. Because all events of a pattern are assigned to the same room, this enforces the contiguous room stability requirement.

Finally, let P_{rt} denote the set of all patterns which include an event in time period t, and for which room r is suitable, i.e. $P_{rt} = \{p \in P\colon r \in R_p, t \in T_p)\}$.

2.3. Integer Programming Formulation

Using the notation defined in Section 2.2, we present an integer programming formulation of a pattern-based set packing model for room assignment. In this formulation, the binary variables x_{pr} are indexed by feasible pattern-to-room assignments. Specifically, let the variable x_{pr} take the value 1 if pattern $p \in P$ is to be held in room $r \in R_p$. For a given objective function w (representing some measure of solution quality), an optimal assignment of patterns to rooms can be determined by solving the following integer programme (1)–(5).

$$
\begin{aligned}
\text{maximise} \quad & \sum_{p \in P} \sum_{r \in R_p} w_{pr} x_{pr} && && (1)\\
\text{subject to} \quad & \sum_{p \in P_{rt}} x_{pr} \leq 1, && r \in R,\ t \in T_r && (2)\\
& \sum_{p \in P_e} \sum_{r \in R_p} x_{pr} \leq 1, && e \in E && (3)\\
& \sum_{p \in P_c} x_{pr} \leq 1, && c \in C,\ r \in R_c && (4)\\
& x_{pr} \in \{0, 1\}, && p \in P,\ r \in R_p && (5)
\end{aligned}
$$

Constraints (2) ensure that at most one event is assigned to each room in each period, while constraints (3) ensure that at most one room is assigned for each event. Constraints (3) do not need to be met with equality, because it is not assumed that a feasible room assignment for all events will exist. Constraints (4) ensure that each course uses at most one pattern per room, i.e. all events from a course that are assigned to a room, should be part of the same (maximal) pattern.

The model is solved in a hierarchical or lexicographic manner i.e. successively for a prescribed series of solution quality measures. This means that one model is solved for each of the different objectives, where each objective function appears as a hard constraint in subsequent optimisations. The particular objectives used and their lexicographic ordering will depend on the needs of a particular institution. For example, given the objective functions and their values (w^l, w^l_0), $l \in \{1, \ldots, k-1\}$, the kth integer programme would include constraints (6).

$$
\sum_{p \in P} \sum_{r \in R_p} w^l_{pr} x_{pr} \geq w^l_0, \qquad l \in \{1, \ldots, k-1\}. \tag{6}
$$

The model is referred to as pattern-based because P contains all patterns of events for each course. However, depending on the objective function w, we can formulate the model with a restricted set of patterns $\bar{P} \subseteq P$ without losing modelling power.

If $\bar{P}$ is restricted to only the patterns which correspond to a single event, i.e. $\bar{P} = E$, then the *event-based* model is obtained. This can be used for any measure of solution quality which relates to the suitability of a room for a particular event i.e. event-based measures. This is in contrast to pattern-based measures which relate to the suitability of a room for any set of course events (see Section 2.4).

For an event-based model, if we consider the additional requirement of contiguous room stability, then for each long event we must include the pattern of all constituent events together, and remove the single-event pattern for each of the long events. This is no longer a purely event-based model, which has implications for its complexity and computational difficulty, as explained in Section 3. For purely event-based models, and those which require contiguous room stability, we must omit constraints (4) which are only valid when an event can be part of more than one pattern.

If $\bar{P}$ is restricted to only those patterns corresponding to a complete course, i.e. $\bar{P} = C$, then the *course-based* model is obtained. Note that any feasible solution to the course-based model requires that each course uses the same room for all events, which is not usually feasible in practice. Constraints (4) should again be omitted, as they are redundant in this case.

2.4. Measures of Solution Quality

There are many, sometimes conflicting, measures of solution quality which can be either event- or pattern-based. We define several common quality measures which are all event-based, with the exception of *course room stability* which is pattern-based. If we need to optimise or constrain a pattern-based measure (as in constraint (6)), a pattern-based model is required. Event-based measures, however, can apply to either an event- or pattern-based model. Note that each event-based objective coefficient includes the term $length_p$, which provides linear scaling for when p contains more than one event.

Several measures of solution quality are described below, and defined in (7)–(12) for the coefficients w_{pr} in the objective function (1).

Event hours (EH). Maximise the total number of events assigned a room over all events. If it is known that a feasible room assignment exists, this is equal to the total number of events in E, and this quality measure can be omitted. Furthermore, in this case, an explicit lexicographic constraint (6) is not required in subsequent iterations, because constraints (3) can be treated as equalities which has the same effect.

$$w_{pr} = length_p, \qquad p \in P,\ r \in R_p \tag{7}$$

Seated student hours (SH). Maximise the total number of hours spent by students in all events assigned a room i.e. events are weighted by their number of students. This is only used when it is not possible to assign a room for all events, and we wish to prioritise events of large courses to be assigned.

$$w_{pr} = length_p * size_c, \qquad c \in C,\ p \in P_c,\ r \in R_p \tag{8}$$

Seat utilisation (SU). Maximise the total ratio of the number of students to the room size over all events assigned a room. This is only used when it is not possible to assign a room for all events, and we wish to prioritise a close fitting of events into rooms.

$$w_{pr} = length_p * \frac{size_c}{size_r}, \qquad c \in C,\ p \in P_c,\ r \in R_p \tag{9}$$

Room preference (RP). Maximise the total course-to-room preference over all events assigned a room. This may be a teacher's preference, or it may be used to teach courses close to the relevant teaching department's offices, as at the University of Auckland. Preferences are determined at the department-to-building level (i.e. all courses from each department have the same preference for all rooms from each building) and may take the value -1, 0 or 1 to indicate undesirability, indifference, or preference.

$$w_{pr} = length_p * Pref(c, r), \qquad c \in C,\ p \in P_c,\ r \in R_p \tag{10}$$

Course room stability (RS). Minimise the total number of different rooms, assigned to each course, over all courses. The disruption to the room stability of a course by one of its patterns is given by $(length_c - length_p)/length_c$. In a feasible room assignment, the sum of these fractions by patterns of a course will sum to the number of additional rooms used, relative to the target '1 room per course'. For example, a course with 3 events could use just 1 pattern (all events in the same room), 2 patterns (2 events in the same room, 1 in a different room), or 3 patterns (each event in a different room). Using the disruption formula, the first case with 1 pattern causes a disruption of zero since no additional rooms are used. The second case will cause a disruption of 1/3 for the larger pattern, and 2/3 for the smaller pattern, summing to 1 additional room. The 3 patterns of the final case disrupt stability by 2/3 each, summing to 2 additional rooms.

$$w_{pr} = -\frac{length_c - length_p}{length_c}, \qquad c \in C, p \in P_c, r \in R_p \tag{11}$$

Spare seat robustness (SR). Maximise the total robustness of the room assignment to changes in each course's enrolment size, $size_c$. Because the room assignment is typically decided prior to student enrolment, $size_c$ is necessarily an estimate of the number of students who will enrol. Therefore, a room which is close in size to the expected enrolment of an assigned course may be considered non-robust to variability in the enrolment size. In practice, the enrolment variability is likely to be different for each course. For example, the enrolment for an entry level course (or one with few pre-requisites) may be less predictable than enrolment for an advanced course on a structured study pathway. An example of a general robustness function is given below, where the room utilisation ($size_c/size_r$) is considered sufficiently robust when below α, and non-robust when above β.

$$w_{pr} = length_p * \begin{cases} 1 & \frac{size_c}{size_r} < \alpha \\ \frac{\beta - \frac{size_c}{size_r}}{\beta - \alpha} & \alpha \leq \frac{size_c}{size_r} < \beta \\ 0 & \beta \leq \frac{size_c}{size_r} \end{cases} \qquad c \in C,\ p \in P_c,\ r \in R_p \tag{12}$$

In this paper we use the parameters of 0.7 for α and 0.9 for β, giving the robustness function shown in Figure 1.

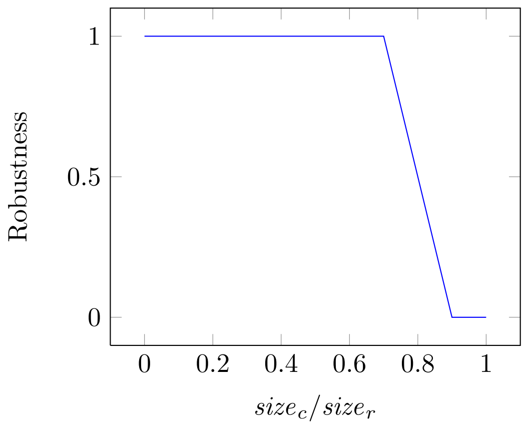

Figure 1: Robustness Function

3. Computational Difficulty

The computational complexity of the room assignment problem is addressed by Carter and Tovey (1992). In this section we review and expand upon their findings, through an insight into the structure of the mathematical programmes.

3.1. Event-based problems

The simplest class of room assignment problems are those which can be formulated with an event-based model ($\bar{P} = E$). This requires that *long* events which span multiple time periods do not need the same room for each period (i.e. no contiguous room stability requirement). It is also assumed that we are not measuring course room stability. Because there are no interdependencies between periods, Carter and Tovey refer to this as the '1-period' problem where each period may be solved separately as an assignment problem. For any objective function (1), the constraint matrix defined by (2)–(3) (since (4) is invalid) of this problem is known to be totally unimodular. Therefore, event-based models can be solved in polynomial time using an assignment problem algorithm (e.g. the Hungarian algorithm), or solving the event-based linear programme.

3.2. Event-based problems with contiguous room stability

A more practically useful class of problems are those which enforce contiguous room stability on long events. Carter and Tovey (1992) refer to this as the 'interval problem' and prove it is NP-hard to find a feasible solution even when the problem is limited to just two time periods. As introduced in Section 2.2, modelling contiguous room stability means we can no longer use a purely event-based model, because patterns are required to place the constituent events of a long event into the same room. This alters the matrix structure, such that fractions can occur i.e. the LP relaxation is no longer guaranteed to be naturally integer. The smallest example of this was presented by Carter and Tovey, shown here as Example 2. For each course c in Table 5, events are shown in their respective time periods, and the feasible rooms for this course are given as R_c. For our formulation defined by (1)–(5), the constraint matrix for this problem is shown in Figure

5. This example happens to also be a course-based problem (one pattern corresponding to all a course's events), so variables are generated for each course for each feasible room. Constraints (2) are identified by the period & room, and constraints (3) are identified by the course they apply to.

Example 2. A minimal event-based problem with contiguous room stability requirements featuring fractionating 7-order 2-cycles

c	t_1	t_2	R_c
A	e_1		r_1, r_2
B	e_1	e_2	r_2, r_3
C	e_1		r_3, r_4
D	e_1	e_2	r_1, r_4
E		e_1	r_1, r_3
F		e_1	r_2, r_4

Table 5: Time Periods and Feasible Rooms

$$\begin{array}{cc}
 & \begin{array}{cccccccccccc} Br_2 & Ar_2 & Ar_1 & Dr_1 & Er_1 & Er_3 & Br_3 & Cr_3 & Cr_4 & Dr_4 & Fr_4 & Fr_2 \end{array} \\
\begin{array}{c} t_1r_2 \\ A \\ t_1r_1 \\ t_2r_1 \\ E \\ t_2r_3 \\ t_1r_3 \\ C \\ t_1r_4 \\ t_2r_4 \\ F \\ t_2r_2 \\ B \\ D \end{array} &
\left[\begin{array}{cccccccccccc}
1 & 1 & & & & & & & & & & \\
 & 1 & 1 & & & & & & & & & \\
 & & 1 & 1 & & & & & & & & \\
 & & & 1 & 1 & & & & & & & \\
 & & & & 1 & 1 & & & & & & \\
 & & & & & 1 & 1 & & & & & \\
 & & & & & & 1 & 1 & & & & \\
 & & & & & & & 1 & 1 & & & \\
 & & & & & & & & 1 & 1 & & \\
 & & & & & & & & & 1 & 1 & \\
 & & & & & & & & & & 1 & 1 \\
1 & & & & & & & & & & & 1 \\
1 & & & & & & 1 & & & & & \\
 & & & 1 & & & & & & 1 & &
\end{array}\right]
\end{array}
\leq
\left[\begin{array}{c} 1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1\\1 \end{array}\right]$$

Figure 2: Set Packing Constraint Matrix

Solving the IP (with constraint matrix shown in Figure 2) to maximise the number of *event hours*, subject to the contiguous room stability requirements, will find an optimal solution which assigns 7 (out of 8) events. However, the LP relaxation is able to assign all 8 events, with each variable taking the value of 0.5.

Early work into the properties of binary matrices (Berge, 1972; Ryan and Falkner, 1988) shows that odd-order 2-cycles (submatrices with row and column sums equal to 2) within a binary constraint matrix permit fractional solutions to occur. Conversely, if no such cycles exist, the matrix is said to be *balanced* and the problem will be naturally integer (i.e. solvable as a linear programme). The rows and columns in the Example 2 constraint matrix (Figure 2) have been ordered to show the odd-order 2-cycles which cause the observed fractional LP solution and corresponding integrality gap. A 7-order cycle can be formed by starting at any of the 'B' or 'D' columns (4 in total), and connecting to each right-adjacent variable until the other column from this course is encountered (treating the right-most column as connected to the left-most). The cycle which starts at Dr_1 is shaded.

In order for this type of cycle (Figure 2) to cause an integrality gap, with respect to the *event hours* objective, a very specific structure must be present. Specifically, each of the six possible combinations of two rooms (out of the four rooms in total) must be the only feasible rooms (R_c) for each of the six courses. Furthermore, there must be no overlap in feasible rooms between the courses featuring in t_1 only, those in t_2 only and those in both periods. Course events can typically be held in any room which provides *at least* enough seats for the course size, and possesses *at least* the requested room attributes. Therefore, the set of feasible rooms for a course will be a superset of those for any larger course and for any course requiring additional attributes. Because of this nested set relationship, it is unlikely that so many different combinations of rooms will occur as the set of feasible rooms for different courses. Furthermore, t_1 and t_2 must be consecutive time periods, rather than any two time periods. Any alteration to the feasible rooms or time periods for each course will close the integrality gap and potentially break the cycle structure in the matrix. For example, if room 1 was removed from the set of feasible rooms for course A, this would break the cycle shown in Figure 2, and the optimal LP solution would have an objective value of 7, the same as the optimal IP solution. Conversely, if room 3 was added to the set of feasible rooms for course A (as well as existing rooms 1 and 2), an IP solution would exist at the LP objective value of 8, again closing the integrality gap.

It is also possible to construct higher order cycles by either extending this cycle (Figure 2) through more courses within the 2 time periods, or by extending across more contiguous time periods. However, these rely on even greater specific structure to be present in the problem.

When the fractional solutions corresponding to odd-order cycles do not cause an integrality gap, they are not precluded from appearing in a solution to the LP, however they are less likely to be found by an IP solver. This was confirmed in our tests on data from the University of Auckland (for the event-based model with contiguous room stability) for all objectives listed in Section 2.4. Solving the LP relaxation returns a solution with a very small number of fractional variables (typically corresponding to 1 or 2 sets of cycles shown in Figure 2), with no integrality gap (for any objective). Interestingly, if we solve the IP with Gurobi (Gurobi Optimization, Inc., 2013), a proprietary solver, an integer optimal solution is found at the root node even with all presolve, cuts and heuristics disabled. This suggests that Gurobi is performing additional 'integerising' LP iterations when it is solving the LP relaxation of an IP.

Although our problem from the University of Auckland is clearly not naturally integer, the fractions which arise are very limited in number, and do not cause an integrality gap. Without using an IP solver, we were able to find an optimal integer solution by adding small perturbations to the objective coefficients of the patterns representing long events. We were also able to use a cutting plane approach to find an optimal integer solution, by applying any violated odd hole inequalities at the LP optimal solution, and then continuing solving the LP. The results from using the Gurobi IP solver, and from using these LP-based methods, demonstrate that although this optimisation problem is NP-hard, the structure of our practical problems is such that any fractions which arise can be easily handled.

We believe that the improbability of encountering cycles of the nature shown in Figure 2, also explains why the earlier work of Gosselin and Truchon (1986) reported naturally integer LPs. Both our tests and theirs were performed on real data, and branch-and-bound has not been required. Therefore, it seems likely that practical event-based problems with contiguous room stability requirements can be solved efficiently.

3.3. Pattern-based problems

The most difficult class of room assignment problems are those which require a pattern-based model ($\bar{P} = P$), because they consider a pattern-based quality measure such as *course room stability*. We address course room stability as a quality measure to be maximised, rather than a hard constraint where all events of a course must be held in the same room. Carter and Tovey (1992) address only the latter case, which they refer to as the 'non-interval problem'. In the context of our formulation, the non-interval problem is represented by a course-based model, a special case of the pattern-based model, which is unlikely to have a feasible solution for practical problems. To model course room stability as a quality measure, for each course we must generate a pattern for each subset of course events (as per Section 2.2), for each room. For courses with only two events, the three patterns generated per room correspond to the first event only ('pattern 1'), the second event only ('pattern 2'), and both events ('pattern 3').

A minimal example of a difficult pattern-based problem is shown as Example 3. For this problem (specification in Table 6) it is not possible to offer a stable room to all courses. For our formulation defined by (1)–(5), the fractionating part of the constraint matrix is shown in Figure 3. Note that only constraints (2) and variables which relate to 'pattern 3' (both events) for each course are shown.

Example 3. A minimal pattern-based problem featuring fractionating 3-order 2-cycles

c	t_1	t_2	t_3	R_c
A	e_1	e_2		r_1, r_2
B		e_1	e_2	r_1, r_2
C	e_1		e_2	r_1, r_2

Table 6: Time Periods and Feasible Rooms

$$
\begin{array}{c}
 \\ t_1r_1 \\ t_2r_1 \\ t_3r_1 \\ t_1r_2 \\ t_2r_2 \\ t_3r_2 \\ \ldots
\end{array}
\begin{array}{c}
\begin{array}{ccccccc} Ap_3r_1 & Bp_3r_1 & Cp_3r_1 & Ap_3r_2 & Bp_3r_2 & Cp_3r_2 & \ldots \end{array} \\
\left[\begin{array}{ccccccc}
1 & & 1 & & & & \\
1 & 1 & & & & & \\
 & 1 & 1 & & & & \\
 & & & 1 & & 1 & \\
 & & & 1 & 1 & & \\
 & & & & 1 & 1 & \\
 & & & & & &
\end{array}\right]
\end{array}
\leq
\begin{array}{c}
 \\
\left[\begin{array}{c} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ \ldots \end{array}\right]
\end{array}
$$

Figure 3: Set Packing Partial Constraint Matrix

Solving the IP (with partial constraint matrix shown in Figure 3) to maximise the course room stability, will find an optimal solution with quality of -1 (a penalty of 1). However, the LP relaxation is able to find an optimal solution with quality of 0 (no penalty), with each 'pattern 3' variable (those shown in Figure 3) taking the value 0.5, and other variables taking the value 0.

Here, integer solutions incur a penalty because they require at least one of the courses to use the undesirable 'pattern 1' and 'pattern 2' variables, which assign two events from the same course to different rooms. The constraint matrix in Figure 3 shows how the desirable 'pattern 3' variables of each course can form odd-order 2-cycles (as shaded). Note that the cycle is entirely contained within constraints (2), meaning that two variables need only be connected by both representing patterns occupying the same room at the same time. This is unlike the cycles in Example 2 which also involve constraints (3).

The requirements for this type of cycle (Figure 3) to exist are that there must be three courses which share two common feasible rooms and each course must feature in a different two of the three time periods. To cause an integrality gap with respect to the course room stability objective, there must also be a relative shortage of available feasible rooms for these courses, in the particular time periods. This could be due to a generally high utilisation rate over all rooms, or because particular sizes and types of rooms are in shortage. Clearly, if a third room was introduced into Example 3 which was feasible for even one of the courses, the same cycles would exist, yet there would no longer be an integrality gap.

Because there is no specific room feasibility requirement (unlike Example 2), and the cycles can be formed over *any* three time periods, there are many more opportunities for such cycles to occur. When these cycles are part of a larger problem, note that the courses do not need to have the same number of events as one another, because the pattern-based model allows any subset of events to be independent (room-wise) from the rest of a course's events. As a result, courses with a large number of events are a major contributor to this type of fractionality, as they introduce many patterns which span different time periods. Larger cycles with this basic structure can also exist, e.g. using 5 courses and 5 time periods instead of 3.

We solved the LP relaxation for the pattern-based model optimising course room stability on data from the University of Auckland and the ITC. The LP solutions were typically much more fractional than those for the event based model with contiguous room stability, and most problems with a non-zero penalty at the IP optimal solution had an integrality gap. A cutting plane approach using odd-hole and clique inequalities (mentioned for Example 2) was much less effective due to many opportunities for the fractions to re-occur without reducing the gap. Consequently, most practical pattern-based problems require the use of a sophisticated IP solver (utilising techniques such as presolve, cuts, heuristics and branching), as covered in our main results in Section 5.

3.4. Lexicographic optimisation constraints

So far we have considered the difficulty of room assignment problems defined by (1)–(5). The remaining factor to address is the effect of adding lexicographic optimisation constraints (6). When solving a purely event-based model, recall that the constraint matrix (defined by (2)–(3), since (4) is invalid) is totally unimodular. As a consequence, the polytope defined by the constraints has integer extreme points. If we solve this model to optimality for an event-based objective measure, the solution must lie on a facet of the polytope, which itself must have integer extreme points. Therefore, if we add a lexicographic constraint (6) to an event-based model, the new feasible region is this face, which must remain naturally integer. Although the new constraint matrix may no longer be totally unimodular (due to the elements of constraint (6)), it will retain the naturally integer property for any number of constraints added through this process. The LP relaxation may be slightly more difficult to solve for each lexicographic constraint added, however no integer programming is required, so the solve time should be acceptable for all practical problems.

For event-based models with contiguous room stability requirements, and for pattern-based models, we have demonstrated that fractional extreme points exist on the polytope. Adding a lexicographic constraint will only limit the feasible region to a facet of this polytope, which may still include these extreme points. Therefore, adding lexicographic constraints will not (necessarily) make the problem easier or remove fractional solutions. However, these two models differ in the quantity and nature of fractional solutions which appear for practical instances. Due to the limited fractionating structures in the LP relaxation for event-based models with contiguous room stability requirements, these can typically still be solved in an acceptable time.

As shown by Example 3, fractionating structures form more readily in pattern-based models, which can cause them to be significantly more difficult to solve than event-based models. Also, in a lexicographic ordering of objectives, once a pattern-based measure is used, all subsequent iterations will require a pattern-based model. Because of the fractionating potential, the solve time for different pattern-based problems can vary substantially, which is the main focus of Section 5, our practical computational results.

4. Course Timetabling at the University of Auckland

In this section we outline the process which was successfully used to find feasible course timetables at the University of Auckland, and optimal classroom assignments to those timetables, during the years 2005 to 2010.

4.1. The Problem

The academic calendar at the University of Auckland primarily consists of two twelve-week semesters. The first semester begins at the start of March, while the second begins in mid-July. In each semester, a weekly teaching timetable is repeated which spans the fifty core teaching hours, from 08:00 to 18:00, Monday through Friday. Teaching departments within the university offer courses, which are often part of one or more *curricula*, a set of courses taken by a common cohort of students simultaneously. Because timetable clashes are based on the curricula (rather than student enrolment data), this problem is a Curriculum-Based Course Timetabling (CB-CTT) problem.

The main City Campus has a pool of common teaching space or *pool rooms* which can be utilised in each hour of each weekday. The Lecture Theatre Management Unit (LTMU) are responsible for the assignment and booking of these rooms, both ad hoc and for timetabled teaching. Statistics relating to pool room requests on the City Campus in Semesters 1 and 2 of 2010 are given in Table 7. For the purposes of Table 7, an "event" refers to a class meeting of any length (i.e. including *long* events).

2010 Semester	1	2
Faculties	10	10
Departments	73	73
Courses	985	911
Events	1965	1866
Total event hours	2383	2231
1 hour events	1561	1516
2 hour events	390	335
3 hour events	14	15
Rooms	72	72

Room attributes
Demonstration Bench
Dual Data Projectors
Dual Image PC
Dual Slide Projectors
Fixed Tier Seating
Moveable Seating
Grand Piano
Radio Microphone
Science Displays
Total Blackout
Wheelchair Access

Table 7: Statistics relating to pool room requests at the University of Auckland City Campus in Semesters 1 and 2 of 2010.

Note that in order to model our practical problem, we need to consider "courses" (as defined by a faculty) which include a lecture component and a tutorial and/or lab component. However, events for these components will most likely be different in terms of required room attributes and number of students, and so cannot be part of the same course $c \in C$ by definition. For example, a course may teach three weekly lecture hours, which all students attend, and five tutorial hours of which each student will attend one. It is desirable for all lectures to be held in the same room, however this is not the case for tutorials, as they are each attended by a different group of students. In the notation of Section 2.2, we can model this by creating one course $c \in C$ for the lecture component (with three events requesting a large lecture theatre) and five courses for the tutorial component (each with one event requesting a small tutorial room). This contributes six courses $c \in C$ to the count in Table 7. For the purposes of generating a timetable, the lecture and tutorial components must be in a curriculum together (to avoid time clashes). However, for the purposes of room assignment, the components are entirely independent.

4.2. Solution Process

There are two distinct phases of the course timetabling process at the University of Auckland. Initially, the 'feasibility phase' occurs from July to October of the previous year. During this time, a feasible timetable is constructed for both semesters and tentative room assignments are made based upon requested room sizes and requested room attributes. This is followed by the 'enrolment phase' which runs from November through to the second week of teaching in each semester.

Initially each faculty will generate its own timetable (time periods for each event) to meet its unique requirements. This includes finding a high quality solution for its students and staff, while respecting non-overlapping requirements for courses within a curriculum, and courses which must be taught by a common lecturer. However, this is typically not a major task as it is managed by making incremental changes to the timetable from the previous year. Guidelines are provided to faculties which help to achieve a good spread of events throughout the day and week. Other rules exist, such as 'two-hour events must start on an even hour' and 'events of three or more hours are accepted only by arrangement', because they are seen to disrupt the ability to place regular one hour events into rooms.

These faculty timetables are collated by the LTMU who attempt to find a high quality feasible room assignment. If no feasible room assignment exists, the timetable must be modified such that one can be found. At a meeting chaired by the LTMU, specific conflicts are addressed and faculties adjust their timetables and/or requested room sizes and attributes. This process repeats until a feasible room assignment can be found. An overview of this process is shown in Figure 4.

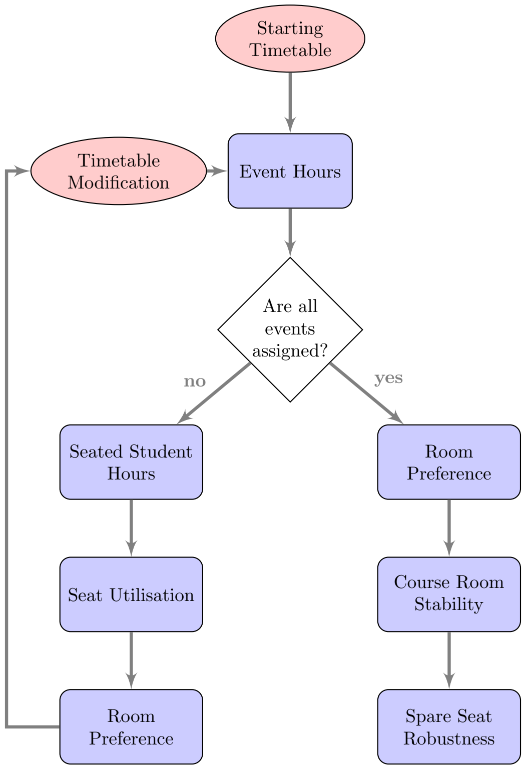

Figure 4: UoA Room Assignment Process

The first room assignment to be solved maximises the the number of *event hours* which can be assigned a room (using an event-based model with contiguous room stability) for the starting timetable. Inevitably, the starting timetable will not permit a feasible room assignment to exist (due to overloading of the most desirable time periods). As mentioned in Section 2, it is useful to lexicographically optimise several other quality measures, which will assist in determining how to modify the timetable. In this case we maximise the *seated student hours*, then *seat utilisation* and finally *room preference* to ensure that the specific events which are assigned to a room tend to be larger events which fit well into their rooms. This lexicographic process will need to occur for each timetable modification, however each of these measures is event-based, so a relatively fast run-time can be expected (see Section 3).

Once all events can be assigned to a room (i.e. the objective value from maximising the *event hours* equals the total number of events), a feasible room assignment has been found. Following this, the most important quality measures for a feasible room assignment are optimised. The first priority is *room preference* which places events into rooms close to their teaching department. Of secondary importance is the *course room stability* which will then attempt to put events of a course into a common room. Finally, optimising *spare seat robustness* is useful to improve the likelihood that this room assignment will remain feasible, withstanding the inevitable variability in enrolment numbers. *Course room stability* is known to be a computationally difficult measure (see Section 3), however this measure only needs to be optimised for feasible room assignments.

After publication of the calendar and feasible timetable for the following year, the enrolment phase begins. While the University is taking enrolments, the numbers are monitored closely, and changes to the timetable and room assignments are considered if necessary.

By involving the faculties directly from the timetable generation through to any arbitration, staff requests are satisfied as much as possible. In their practical study of automating a single department's timetabling system, Schimmelpfeng and Helber (2007) observed that staff appeared to demonstrate knowledge of a timetable which had worked in previous years. Until a powerful automated process exists to generate university timetables for very large institutions (with complex and internally variable quality measures), we believe it is valuable to retain staff involvement as much as possible.

5. Results

All our computational experiments were run using Gurobi 5.1 running on 64-bit Debian 7, with a quad-core 3.33 GHz processor (Intel i5-2500K). To exploit the well-studied structure of set packing problems (Avella and Vasil'Ev, 2005), only zero-half and clique cuts were generated, and both were set to 'aggressive' generation (Gurobi Optimization, Inc., 2013). The time limit was set at 3600 seconds.

5.1. The University of Auckland 2010

To validate our method we process the University of Auckland's timetabling data from Semesters 1 and 2 in 2010. We first test on 'starting' timetables which have been generated by faculties, and for which a feasible room assignment is unlikely to exist. We also test on a 'final' timetable which has been modified (see Section 4.2) such that a feasible room assignment is known to exist. The specific quality measures chosen are those shown in the flowchart from Figure 4, and contiguous room stability is enforced.

		Objectives/ Iterations			
Semester One		EH	SH	SU	RP
Event Hours (total 2400)	EH	2374 *	2374 =	2374 =	2374 =
Seated Student Hours	SH	252211	253864 *	253864 =	253864 =
Seat Utilisation	SU	1769.3	1767.9	2077.6 *	2077.6 =
Room Preference	RP	530	571	722	839 *
Solve Time (s)		0.6	1.9	2.1	5.5
Semester Two		EH	SH	SU	RP
Event Hours (total 2234)	EH	2211 *	2211 =	2211 =	2211 =
Seated Student Hours	SH	234828	237579 *	237579 =	237579 =
Seat Utilisation	SU	1572.5	1572.9	1940.9 *	1940.9 =
Room Preference	RP	466	458	614	727 *
Solve Time (s)		0.5	1.5	1.7	5.1

Table 8: UoA 2010 Starting Timetable Room Assignment

Table 8 shows the results of our method on the starting infeasible timetable from each semester. A column lists the results of solving an iteration of the lexicographic algorithm i.e. solving an integer programme (1)–(5) maximising the objective marked with an asterisk, subject to any lexicographic constraints (6) marked with an equality sign. The first iteration maximises the *event hours* but is unable to find a room for all events, demonstrating that this is an infeasible timetable. This differs from the total number of events listed in Tables 7 & 9, as the latter tables relate to the 'final' timetable, and reflect unrelated changes to planned courses which occur in the interim period.

The solve time for each integer programme is notably low, because these are all event-based solution quality measures. As explained in Section 3, event-based models with contiguous room stability requirements have near-integral LP relaxations. After solving all four models we have a partial room assignment which is Pareto efficient in terms of all objectives. This will provide useful information to assist in modifying the timetable to achieve feasibility in the room assignment.

		Objectives/ Iterations							
Semester One		EH		RP		RS		SR	
Event Hours (total 2383)	EH	2383	*	2383	=	2383	=	2383	=
Room Preference	RP	490		1561	*	1561	=	1561	=
Course Room Stability	RS	-760		-594		-93	*	-93	=
Spare Seat Robustness	SR	1086.1		1205.4		1213.9		1415.5	*
Solve Time (s)		0.6		0.7		256.5		815.7	
Semester Two		EH		RP		RS		SR	
Event Hours (total 2231)	EH	2231	*	2231	=	2231	=	2231	=
Room Preference	RP	463		1435	*	1435	=	1435	=
Course Room Stability	RS	-730		-546		-80	*	-80	=
Spare Seat Robustness	SR	884.5		994.5		1043.0		1312.8	*
Solve Time (s)		0.5		0.7		68.5		166.4	

Table 9: UoA 2010 Final Timetable Room Assignment

Table 9 shows the results of our method on final feasible timetables, yielding a feasible room assignment without any further need for altering the timetable. This is shown by the fact that the first iteration is able to find a room for all events, 2383 and 2231, respectively. Observe that the first two iterations have a short solve time, while the latter two iterations take considerably longer. This is because optimising the *course room stability* uses a pattern-based model which requires the use of integer programming techniques (presolve, cuts, heuristics and branching) to find and confirm an optimal solution.

In theory, many iterations of a lexicographic ("optimise-and-fix") algorithm will eventually tightly constrain the problem. However, in this case we see that significant gains continue to be made to later quality measures, and the solve times remain manageable. While this does give a Pareto optimal solution, the solve times are low enough to suggest there may be enough flexibility to apply more complex multiobjective optimisation methods which generate a "frontier" of many Pareto efficient solutions.

5.2. International Timetabling Competition 2007

As previously stated, the main focus of our work is on practical problems which feature many room-related solution quality measures. However, we also address instances from Track 3 of the 2007 International Timetabling Competition (ITC), as these are widely used in the literature as benchmarks. For details on the competition, the reader is referred to the competition website (ITC, 2007), official documentation (Di Gaspero et al., 2007) and the competition results (McCollum et al., 2010). We particularly recommend a follow up-paper (Bonutti et al., 2012) dedicated to benchmarking in course timetabling, which gives detailed information about the structure of the ITC problems and introduces alternative specifications for measuring timetable quality. These specifications are subject to ongoing development, so we offer a discussion into the potential shortcomings in Section 5.4.

It is firstly noted that benchmarking our work directly against ITC entries is not possible, due to the fact that we focus solely on the room assignment. However, we are interested in finding a room assignment for timetables from the ITC entries, to test the performance of the room assignment model on a diverse set of instances. All timetables were retrieved from the publicly accessible listing at `http://tabu.diegm.uniud.it/ctt` (Bonutti et al., 2008), where our final solutions have also been uploaded.

To address the ITC problems, we first solve for the *UD2* specification (as used in the competition) which treats course room stability as the only room-related solution quality measure. To solve the room assignment, we have used the timetables from Tomáš Müller's heuristically-generated solutions, which were the overall winner of the ITC (Müller, 2009) and are available for the full set of ITC problems.

Problem		Müller's Results		Our Room Results			
Name	Room Util (%)	Timetable	Room	IP	LP	Nodes	Time (s)
comp02	71	35	0	0	0	0	1.0
comp03	63	66	0	0	0	0	0.4
comp04	64	35	0	0	0	0	1.6
comp05	47	294	4	4	4 I	0	0.1
comp06	80	37	0	0	0	0	12.8
comp07	87	6	1	1	0	38491	3600.0
comp08	72	38	0	0	0	0	8.4
comp09	62	100	0	0	0	0	0.9
comp10	82	6	1	0	0	799	82.7
comp11	72	0	0	0	0 I	0	0.2
comp12	55	319	1	1	1	0	0.1
comp13	65	61	0	0	0	0	2.6
comp14	65	53	0	0	0	0	0.4
comp15	63	70	0	0	0	0	0.5
comp16	73	30	0	0	0	0	5.0
comp17	80	70	0	0	0	0	11.3
comp18	43	75	0	0	0 I	0	0.1
comp19	69	57	0	0	0	0	1.7
comp20	82	20	2	1	0	4440	160.9
comp21	73	89	0	0	0	0	4.4

Table 10: Results on Müller's ITC (*UD2*) Timetables

Our results for all ITC problems except *comp01* are shown in Table 10. Column 3 gives the quality (penalty) from the time-related solution quality measures of Müller's solution, which is a sum of several weighted penalty factors defined in Bonutti et al. (2012). Column 4 gives the penalty from room-related solution quality measures from Müller's solution, which is equivalent to the course room stability penalty (for the *UD2* specification). We can compare this to our IP optimal course room stability penalty in column 5. Column 6 gives the objective value of the LP relaxation, where an 'I' represents an integral LP relaxation. Column 7 gives the number of nodes which were explored in the solve process, and column 8 gives the run-time to optimality (or the time limit).

We do not include a result for *comp01*, because our approach does not model a "soft" room size. When constrained to original room sizes, the *comp01* room assignment problem is infeasible for any timetable, as noted by Asín Achá and Nieuwenhuis (2012). In our method, an infeasible room assignment (for a given timetable) is confirmed when the optimal room assignment maximising the *event hours* is not able to assign all events to a room.

Note that three of the problems had integral LP relaxations, and were very quick (<0.5s) to solve. Another fourteen of the problems did not have integral LP relaxations, however were able to find an optimal integer solution at the root node (i.e. without branching). These problems did contain odd-order cycle induced fractions, however Gurobi was able to find an integer solution relatively quickly (<15s) using cuts and/or heuristics. Only one problem, *comp10*, used branching to find an optimal solution when there was no integrality gap.

The remaining two problems, *comp07* and *comp20*, were the only cases of odd-order cycles causing an integrality gap, as demonstrated in Example 3 from Section 3. In these cases, there were many ways for the cycles to re-occur (with the same objective value) after branching or cuts were applied. For *comp20*, the solver was able to prove no integer solution could exist at the LP objective value, while *comp07* required a substantially longer time, exceeding the time limit. Because we used Gurobi's aggressive cut generation parameters, significant computational work was expended on attempting to improve the lower bound to confirm optimality. However, it should be noted that a good (or even optimal) solution can be found more quickly than the time required to confirm optimality, particularly when parameters are chosen for this purpose.

Focussing on the most difficult problems by solve time in our study, it is evident that there is a correlation between the difficulty of the room assignment, and the room utilisation (given in column 2 of Table 10). In this case, utilisation is measured as the total number of events divided by the total number of available time periods for all rooms. The five most difficult problems (*comp07*, *comp20*, *comp10*, *comp06*, *comp17*) are those with the five highest utilisations, all at least 80%. This is consistent with our theoretical results from Section 3, where we demonstrate how problems with a high room utilisation are more likely to exploit the odd order cycles and cause an integrality gap between the IP and LP relaxation. Also, a high number of events per course will "link" more time periods together, such that there are more opportunities for odd-order cycles to occur. The ITC problems have an average of 3.5 events per course, and most problems have a course with 7 or even 9 events.

5.3. International Timetabling Competition Extended Specification

Although the *UD2* specification was used in the ITC, the follow-up/extension by Bonutti et al. (2012) introduces three other specifications which have received significantly less attention in the literature. Here we address *UD5*, as it includes a room-related solution quality measure, *travel distance*, which relates to the physical distance which students within a curriculum must travel between consecutive events.

Problem Name	Shaerf's Results Timetable	Room	Our Room Results IP	Nodes	Time (s)
comp02	128	42	40	0	0.4
comp03	163	28	10	15	1.4
comp04	82	8	2	0	0.6
comp05	606	89	52	0	0.0
comp06	112	36	18	20	7.4
comp07	61	36	16	28	6.5
comp08	77	6	0	0	0.4
comp09	164	12	0	0	0.3
comp10	62	74	30	45	11.8
comp11	0	0	0	0	0.0
comp13	153	16	4	0	0.7
comp14	93	32	16	9	0.8
comp15	168	52	48	0	0.1
comp16	99	30	28	0	1.2
comp17	145	40	36	0	2.2
comp18	122	26	22	0	0.1
comp19	135	18	12	0	0.1
comp21	174	14	4	20	0.7

Table 11: Results on Shaerf's *UD5* Timetables

To solve these problems we have used the timetables from Andrea Shaerf's heuristically-generated solutions (Bellio et al., 2012) for the *UD5* specification. To model this measure of travel time, we needed to add auxiliary variables and constraints to an event-based model. The results are shown in Table 11 (with the same column heading interpretations as Table 10).

Although this extension of the event-based model is no longer naturally integer, the results show rapid solve times (column 6). We are able to improve on existing room assignment solutions (column 3 vs column 4) in every case where a penalty is incurred, often by a substantial margin. As with the tests for the *UD2* specification, we have used the original room size limits rather than the modified limits from Shaerf's solutions. This allows us to avoid incurring any penalty for exceeding the size of the room. However, inevitably some problems (*comp12* and *comp20*) have no feasible room assignment for the given timetable, without expanding the room size.

Finally, the online listing contains the best solutions and best bounds found for the *UD2* problems from any method, with no restrictions on run-time (Bonutti et al., 2008). The majority of best known solutions incur no room stability penalty, and we are able to generate an equivalent room assignment quickly. However, the previously best known solution to *comp21* by Moritz Mühlenthaler incurred a timetable penalty of 74, and a room assignment penalty of 1 (for a total penalty of 75). For this timetable, our model was able to find a room assignment with 0 penalty after 30.4 seconds of run-time, yielding a new best solution with a total penalty of 74. The lower bound of 74, which was provided by Gerald Lach, confirms our result is an optimal solution to *comp21*. This result is encouraging in terms of validating the co-utilisation of both heuristic methods (as used by Mühlenthaler) and optimisation methods for difficult problems.

5.4. Comments on ITC Datasets

Finally, we would like to discuss potential shortcomings of the way the ITC problems are designed, in terms of the problem structure and the way quality is quantified. Although the problems have been derived from real data at the University of Udine in Italy, we find some features to be unusual. We are particularly interested in how these widely-used benchmark instances relate to practical problems.

Firstly, we find that courses in the ITC problems frequently have an extremely high number of events. Most problems have several courses with up to 7 events. In our experience, it is uncommon for a course to need to hold more than 4 events in the same week, all desiring to be in the same room. For example, at the University of Auckland, normal size courses typically have 3 lectures and 1 tutorial per week. However, because these components are usually treated as separate courses in the model, the largest course $c \in C$ comprises only the 3 lecture events.

The utilisation of university resources is another factor which appears to be abnormally high in the ITC problems. This naturally makes the problems more difficult, as the algorithms operate with less flexibility for placements of events. Studies into the utilisation of teaching space at real universities (Beyrouthy et al., 2007) suggest that rooms are occupied 50% of the time on average, rather than the 60%-80% (see Table 10), which is typical for the ITC problems.

We also find that the scale of ITC problems varies between small to medium size problems, but does not cater to problems faced by the very largest institutions. The largest ITC problem (*comp07*) features 131 courses with 434 events and 20 rooms, which is significantly smaller than the problem faced by the University of Auckland, as shown in Table 7.

As far as solution quality measures are concerned, we find that using a soft limit for room capacity (which features in all five specifications), is less realistic than a hard limit. The majority of rooms will have a certain number of fixed seats which cannot easily be increased, providing a natural hard limit. In the case of the University of Auckland, the number of students cannot legally exceed the number of seats. The "soft" undesirability of a near-full room can be modelled as an event-based solution quality measure similar to spare seat robustness.

For the *UD5* specification addressed in Table 11, the quantification of the *travel distance* penalty is also unusual. A penalty is applied when consecutive events from the same curriculum are held in different buildings. However, the penalty is applied for each curriculum the events feature in. Because pairs of courses may exist together in more than one curriculum, the penalty for a particular set of events is multiplied by the number of curricula they both appear in. This weighting is arbitrary, particularly because the problems include redundant curricula which are *dominated* by other curricula i.e. they feature a subset of the courses of another curriculum. These dominated curricula have no effect on any constraint or quality measure, except to alter the quantification of the travel distance penalty. Potentially the travel distance penalty could be weighted by the number of students influenced, or the distance between buildings (for a problem with more detailed data).

Finally, we would like to discuss the specific choices of quality measures. It is acknowledged by the competition organisers (Bonutti et al., 2012), and many researchers in the field, that there is no universal measure of timetable quality. Not only do different rankings of importance of commonly-desired timetable features exist, but there can even be contradicting views of whether a given feature is desirable. For example, two ITC quality measures relating to curriculum compactness (which favour a "bunching" of events) may be considered undesirable by some timetablers, who prefer a wider spread of events throughout the day. Furthermore, even for the same set of priorities there may be many equally valid ways to define or quantify a quality measure in practice. As mentioned by Burke et al. (2010), if there are many similar rooms in one building or location, it may be more important to hold all events of a course in one of these rooms rather than measuring stability with respect to a specific room. We are inclined to agree, and note that our results demonstrate why optimising room preference (an event-based measure) is significantly easier than optimising course room stability (which is pattern-based). In our approach, the first priority for a feasible timetable is maximising room preference, which can be solved efficiently. Course room stability is then improved, but only without reducing the total room preference. It is also likely that maximising room preference will implicitly minimise students' travel distance, since courses within a curriculum are typically taught by the same department or faculty.

We use this example of measuring course room stability to demonstrate how different quantifications of quality may be equivalent from a practical perspective, yet differ substantially in difficulty when solving the problem with a particular approach. This is consistent with the sentiment of the ITC competition organisers, that although an algorithm outperforms another on a certain set of benchmarks, this does not imply that it is a superior algorithm in general (McCollum et al., 2010).

6. Concluding Remarks and Future Directions

We have introduced a novel pattern-based formulation for room assignment problems, that generalises the existing models of interval and non-interval scheduling from Carter and Tovey (1992). Most importantly, we have shown how this model can be part of a practical process for full size university timetabling. We are able to solve an exact integer programming model for room assignment quickly enough to get a Pareto optimal solution with respect to several solution quality measures on data from the University of Auckland. We are also able to identify the situations where fractional solutions can arise in the LP relaxation, causing the problems to become more difficult and require greater use of integer programming techniques.

Our approach has also been applied to the ITC problems. We demonstrate that it is possible to improve on many of the heuristically-generated solutions using an exact approach to the room assignment part of the problem. We hope this study helps demonstrate that mathematical programmes can be useful to incorporate into a heuristic framework.

To continue this work, we are interested in implementing more sophisticated multi-objective optimisation methods, which will allow us to explore the trade-offs between objectives more fully. We are also exploring more advanced integer programming techniques to exploit the structure of the most difficult pattern-based problems.

To complement the classroom assignment method presented in this paper, we would like to develop algorithms for automating the timetable modification process. Whether incorporated at the planning stage or post-enrolment stage of university timetabling, timetable modifications are commonly required in practice and have not yet been comprehensively studied.

Acknowledgements

The authors are grateful to Alex Bonutti, Luca Di Gaspero and Andrea Shaerf from the University of Udine for creating and maintaining the online listing of all ITC instances and best-known solutions. We would also like to thank Paul Ketko and Sue Wightman from the University of Auckland Lecture Theatre Management Unit for their assistance with acquiring and processing the data.

Finally, we appreciate the insightful comments of the three anonymous reviewers who have helped to improve this paper.

References

Asín Achá, R., Nieuwenhuis, R., 2012. Curriculum-based course timetabling with SAT and MaxSAT. Annals of Operations Research February 2012, 1–21.

Avella, P., Vasil'Ev, I., 2005. A computational study of a cutting plane algorithm for university course timetabling. Journal of Scheduling 8 (6), 497–514.

Bellio, R., Di Gaspero, L., Schaerf, A., 2012. Design and statistical analysis of a hybrid local search algorithm for course timetabling. Journal of Scheduling 15 (1), 49–61.

Berge, C., 1972. Balanced matrices. Mathematical Programming 2 (1), 19–31.

Beyrouthy, C., Burke, E. K., Landa-Silva, D., McCollum, B., McMullan, P., Parkes, A. J., 2007. Towards improving the utilization of university teaching space. Journal of the Operational Research Society 60 (1), 130–143.

Bonutti, A., De Cesco, F., Di Gaspero, L., Schaerf, A., 2012. Benchmarking curriculum-based course timetabling: formulations, data formats, instances, validation, visualization, and results. Annals of Operations Research 194 (1), 59–70.

Bonutti, A., Di Gaspero, L., Schaerf, A., 2008. Curriculum-based course timetabling (website). Retrieved April 2013, `http://satt.diegm.uniud.it/ctt/`.

Burke, E. K., Mareček, J., Parkes, A. J., Rudová, H., 2008. Uses and abuses of MIP in course timetabling. Poster at the workshop on mixed integer programming, MIP2007, Montréal, 2008, available online at `http://cs.nott.ac.uk/jxm/timetabling/mip2007-poster.pdf`.

Burke, E. K., Mareček, J., Parkes, A. J., Rudová, H., 2010. Decomposition, reformulation, and diving in university course timetabling. Computers & Operations Research 37 (3), 582–597.

Carter, M. W., 1989. A Lagrangian relaxation approach to the classroom assignment problem. INFOR. 27 (2), 230–246.

Carter, M. W., Tovey, C. A., 1992. When is the classroom assignment problem hard? Operations Research 40, S28–S39.

Dammak, A., Elloumi, A., Kamoun, H., 2006. Classroom assignment for exam timetabling. Advances in Engineering Software 37 (10), 659 – 666.

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 2007/08/01, University of Udine DIEGM, Udine, Italy.

Ehrgott, M., 2005. Multicriteria optimization, 2nd Edition. Springer Berlin.

Elloumi, A., Kamoun, H., Jarboui, B., Dammak, A., 2014. The classroom assignment problem: Complexity, size reduction and heuristics. Applied Soft Computing 14, Part C, 677 – 686.

Glassey, C. R., Mizrach, M., 1986. A decision support system for assigning classes to rooms. Interfaces 16 (5), 92–100.

Gosselin, K., Truchon, M., 1986. Allocation of classrooms by linear programming. Journal of the Operational Research Society 37 (6), 561–569.

Gurobi Optimization, Inc., 2013. Gurobi Optimizer Reference Manual. `http://www.gurobi.com`.

ITC, 2007. The Second International Timetabling Competition. Retrieved April 2013, `http://www.cs.qub.ac.uk/itc2007/`.

Lach, G., Lübbecke, M. E., 2008. Optimal university course timetables and the partial transversal polytope. In: McGeoch, C. C. (Ed.), Proceedings of the 7th International Conference on Experimental Algorithms. WEA'08. Springer Berlin, pp. 235–248.

Lach, G., Lübbecke, M. E., 2012. Curriculum based course timetabling: new solutions to Udine benchmark instances. Annals of Operations Research 194 (1), 255–272.

McCollum, B., Schaerf, A., Paechter, B., McMullan, P., Lewis, R., Parkes, A. J., Di Gaspero, L., 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.

MirHassani, S., Habibi, F., 2013. Solution approaches to the course timetabling problem. Artificial Intelligence Review 39 (2), 133–149.

Mirrazavi, S., Mardle, S., Tamiz, M., 2003. A two-phase multiple objective approach to university timetabling utilising optimisation and evolutionary solution methodologies. Journal of the Operational Research Society 54 (11), 1155–1166.

Müller, T., 2009. ITC2007 solver description: a hybrid approach. Annals of Operations Research 172 (1), 429–446.

Nemhauser, G. L., Wolsey, L. A., 1988. Integer and combinatorial optimization. Vol. 18. Wiley New York.

Qualizza, A., Serafini, P., 2005. A column generation scheme for faculty timetabling. In: Burke, E. K., Trick, M. A. (Eds.), Practice and Theory of Automated Timetabling V. Vol. 3616 of Lecture Notes in Computer Science. Springer Berlin, pp. 161–173.

Ryan, D. M., Falkner, J. C., 1988. On the integer properties of scheduling set partitioning models. European Journal of Operational Research 35 (3), 442–456.

Schimmelpfeng, K., Helber, S., 2007. Application of a real-world university-course timetabling model solved by integer programming. OR Spectrum 29 (4), 783–803.

van den Broek, J., Hurkens, C., Woeginger, G., 2009. Timetabling problems at the TU Eindhoven. European Journal of Operational Research 196 (3), 877–885.