Ann Oper Res manuscript No.
(will be inserted by the editor)

A Branch-and-cut Procedure for the Udine Course Timetabling Problem

Edmund K. Burke[1], **Jakub Mareček**[12⋆],
Andrew J. Parkes[1], **Hana Rudová**[2]

[1] Automated Scheduling, Optimisation and Planning Group
School of Computer Science, The University of Nottingham
Nottingham NG8 1BB, UK
e-mail: {jxm,ajp,ekb}@cs.nott.ac.uk
[2] Faculty of Informatics, Masaryk University
Botanická 68a, Brno 602 00, The Czech Republic

Received: July 9, 2010

Abstract A branch-and-cut procedure for the Udine Course Timetabling problem is described. Simple compact integer linear programming formulations of the problem employ only binary variables. In contrast, we give a formulation with fewer variables by using a mix of binary and general integer variables. This formulation has an exponential number of constraints, which are added only upon violation. The number of constraints is exponential, however, only with respect to the upper bound on the general integer variables, which is the number of days per week in Udine Course Timetabling.

A number of further classes of cuts are also introduced, arising from: Enumeration of event/free-period patterns; bounds on the numbers of days of instruction; the desire to exploit integrality of the objective function value; the graph colouring component; and also from various implied bounds. An implementation of the corresponding branch-and-cut procedure is evaluated on the instances from Track 3 of the International Timetabling Competition 2007. Using IBM/ILOG CPLEX it is possible to find provably optimal solutions to two instances (comp01 and comp11) within 15 minutes, and good lower bounds for several other instances within two hours.

Key words integer programming, branch-and-cut, cutting planes, soft constraints, educational timetabling, university course timetabling

⋆ Contact e-mail: jakub@marecek.cz

1 Introduction

There has recently been a considerable interest in curriculum-based university course timetabling. This is largely due to the inclusion of a benchmark problem in the field, Udine Course Timetabling, in Track 3 of the Second International Timetabling Competition [McCollum et al., 2010, Gaspero and Schaerf, 2006, Bonutti et al., 2011]. The Udine Course Timetabling problem consists of an extension of a pre-colouring with a bounded number of uses of each colour [Burke et al., 2004], with four additional complex soft constraints. The goal is to find a feasible bounded colouring, minimising the number of violations of the soft constraints. The soft constraints place emphasis on suitability of rooms with respect to their capacities, suitability of the spread of the events of a course within the weekly timetable, minimisation of the number of distinct rooms each course uses, and most importantly, desirability of various patterns in distinct individual daily timetables. This seems to represent a reasonable trade-off between extensibility to real-world applications, given by the inclusion of some complex soft constraints, and the ease of implementation of research prototypes of solvers, given by the omission of a large number of other constraints. Modelling of these four soft constraints entails considerable increase in the dimensions of the model, making even modest real-life instances difficult to solve using a stand-alone integer programming solver [Burke et al., 2008, 2007], or standard branch-and-cut procedures [Avella and Vasil'ev, 2005].

In this paper, we present a new branch-and-cut procedure, where a number of cuts, exponential only in the number of periods per day, have to be considered to guarantee optimality. The problem and the instances we focus on are introduced in more detail in Section 2. An outline of the integer programming formulation we use is given in Section 3. In Section 4, we present both original problem-specific cuts and applications of well known cuts from graph colouring. An implementation is described in Section 5. Its computational evaluation is provided in Section 6. Finally, related work is discussed in Section 7.

2 Problem Description

Timetabling problems generally require a feasible bounded colourings of conflict graphs, where vertices represent events. There is an edge between two vertices of the conflict graph, if the corresponding events cannot take place at the same time, for instance because some students want to attend both the corresponding events or because those are taught by the same person. An assignment of colours represents assignment of periods, and the bound on the number of uses of a colour represents the limited number of rooms available. (See Welsh [1967] or Burke et al. [2004] for more.) This graph colouring component is often accompanied by a number of soft constraints, and the objective is to minimise the total number of their violations. In university course timetabling [Bardadym, 1996, Carter

and Laporte, 1997, Burke et al., 1997, Schaerf, 1999, Burke and Petrovic, 2002, Petrovic and Burke, 2004, McCollum, 2006], these soft constraints often stipulate that:

- Events should be timetabled in rooms of sufficient sizes
- At least (or at most) a certain number of days of instructions should be timetabled for each group of students and each teacher
- Daily timetables of students or teachers should not exhibit specified patterns. For instance, a single event per day or long gaps in a daily timetable, or on the other hand, six events per day with no gap around lunch time may be deemed undesirable.

The particulars vary widely from university to university. In this paper, we study the Udine Course Timetabling problem, which is maintained by Gaspero and Schaerf [2003], Bonutti et al. [2011] at *Università degli studi di Udine*. There are two important assumptions:

- Events are partitioned into disjoint subsets, called courses; events of any one course have to take place at different times and are all attended by the same number of students
- A small number of distinct, possibly overlapping, sets of courses, representing enrolments prescribed to various groups of students, are identified and referred to as curricula.

Due to the second assumption, this problem is often referred to as "curriculum-based timetabling", as opposed to "student enrolment based timetabling", which tries to minimise the number of conflicts among a possibly large number of distinct student enrolments. The complete input data can be captured by the following:

- C, U, T, R, D, P are sets of courses, curricula, teachers, rooms, days, and periods, respectively
- $\tilde{U}^u$ is the (non-empty) set of courses in curriculum u
- N is a subset of $\mathrm{C} \times \mathrm{P}$, giving forbidden course-period combinations
- E^c is the number of events course c has in a week
- S^c is the number of students enrolled in course c
- M^c is the prescribed minimum number of distinct week-days of instruction for course c
- $\tilde{T}^t$ is the subset of courses C taught by teacher t
- A^r is the capacity of room r
- $\tilde{D}^d$ is the subtuple (ordered subset) of P corresponding to periods in day d
- W is a vector of non-negative weights $(\mathrm{W}^{\mathrm{RCap}}, \mathrm{W}^{\mathrm{TSpr}}, \mathrm{W}^{\mathrm{TCom}}, \mathrm{W}^{\mathrm{RStb}})$ for the four soft constraints that we will describe below.

A formal model of the problem is given in Section 3. Informally, however, the goal is to produce a mapping from events to period-rooms pairs such that:

1. For each course c, exactly E^c events are timetabled

2. No two events take place in the same room in the same period
3. No two events of a single course, no two events taught by a single teacher, and no two events included in a single curriculum are taught at the same time
4. No event of course c is taught in a period p, if $\langle c, p\rangle$ is in N.
5. A linear combination of penalty terms

$$\mathrm{W}^{\mathrm{RCap}}\,\mathrm{P}^{\mathrm{RCap}} + \mathrm{P}^{\mathrm{TSpr}}\,\mathrm{W}^{\mathrm{TSpr}} + \mathrm{P}^{\mathrm{TCom}}\,\mathrm{W}^{\mathrm{TCom}} + \mathrm{P}^{\mathrm{RStb}}\,\mathrm{W}^{\mathrm{RStb}}$$

is minimised, where
   - $\mathrm{P}^{\mathrm{RCap}}$ (for "room capacity") is the number of students left without a seat at an event, summed across all events; this is the difference of the number of students attending an event minus capacity of the allocated room over all events where the difference is positive
   - $\mathrm{P}^{\mathrm{TSpr}}$ (for "spread of events of a course over distinct week-days") sums the difference of the number of prescribed distinct week-days of instruction minus the actual number of distinct week-days of instruction over all courses where the difference is positive
   - $\mathrm{P}^{\mathrm{TCom}}$ (for "time compactness") is the number of isolated events in daily timetables of individual curricula. That is "we account for a violation every time there is one lecture not adjacent to any other lecture on the same day" [Bonutti et al., 2011]
   - $\mathrm{P}^{\mathrm{RStb}}$ (for "room stability") is the number of distinct course-room allocations minus the number of courses.

The original paper of Gaspero and Schaerf [2003] described only four instances of the Udine Course Timetabling Problem of up to 252 events and 57 distinct enrolments, with $(\mathrm{W}^{\mathrm{RCap}}, \mathrm{W}^{\mathrm{TSpr}}, \mathrm{W}^{\mathrm{TCom}}, \mathrm{W}^{\mathrm{RStb}}) = (1, 5, 2, 0)$. Fourteen more instances of up to 434 events and 81 distinct enrolments have now been made available in Track 3 of the International Timetabling Competition, with weights $(\mathrm{W}^{\mathrm{RCap}}, \mathrm{W}^{\mathrm{TSpr}}, \mathrm{W}^{\mathrm{TCom}}, \mathrm{W}^{\mathrm{RStb}}) = (1, 5, 2, 1)$ [Bonutti et al., 2011]. Their dimensions are summarised in Table 1.

Although the instances might seem small, they do pose a challenge to exact solvers. Exact solvers for timetabling problems have been under development since Lawrie [1969] generated feasible solutions to a school timetabling problem using branch and bound and Gomory cuts. Tripathy [1984], for instance, solved a large instance of a school timetabling problem using branch and bound and Lagrangian relaxation. Another milestone is the study of Carter [1989], who solved instances of a course timetabling problem with several soft constraints of up to 287 events using Lagrangian relaxation. More recently, modest instances have even been solved using off-the-shelf solvers [Dimopoulou and Miliotis, 2004, Qualizza and Serafini, 2004, Daskalaki et al., 2004, 2005, Mirhassani, 2006]. For instance, Daskalaki et al. [2004, 2005] solved instances of up to 211 events using ILOG CPLEX, without introducing any user cuts. Al-Yakoob and Sherali [2007] and Schimmelpfeng and Helber [2007] modelled more complex problems, although they have not introduced any constraints penalising

interaction between events in timetables (other than straightforward conflicts), which would have made the problem considerably more difficult. In the most rigorous study so far, Avella and Vasil'ev [2005] presented a branch-and-cut solver for the Benevento Course Timetabling Problem, which forbids some interactions of events in timetables other than conflicts using hard constraints. They have been able to solve instances of up to 233 events and 14 distinct enrolments, but conceded that application of their solver to the four then available instances of Udine Course Timetabling yielded "poor results". Several integer programming formulations of Udine Course Timetabling have been described in Burke et al. [2008, 2007], Lach and Lübbecke [2008, 2011]. On many instances from the International Timetabling Competition 2007, however, the run-time of the linear programming (LP) solver remains a barrier to producing solutions competitive with heuristics based on local search within time limits used

Table 1: Instances of the Udine Course Timetabling problem: numbers of rooms and periods; the number of courses and the sum of their events in a week; frequency, or the portion of period-room slots in use, and utilisation, or the portion of period-seat slots in use [Beyrouthy et al., 2007]; the number of distinct enrolments ("curricula"); numbers of edges and density in conflict graphs (CG) with vertices representing courses, rather than events [Burke et al., 2007], and edges given by curricula and teachers.

Instance	AKA	Rooms	Periods	Courses	Events	Frequency (used slots)	Utilitisation (used seats)	Curricula	Edges in CG (course-based)	Density of CG (course-based)
comp01	Fis0506-1	6	30	30	160	88.89 %	45.98 %	14	53	12.18 %
comp02	Ing0203-2	16	25	82	283	70.75 %	46.28 %	70	401	12.07 %
comp03	Ing0304-1	16	25	72	251	62.75 %	38.30 %	68	342	13.38 %
comp04	Ing0405-3	18	25	79	286	63.56 %	33.22 %	57	212	6.88 %
comp05	Let0405-1	9	36	54	152	46.91 %	43.50 %	139	917	64.08 %
comp06	Ing0506-1	18	25	108	361	80.22 %	45.28 %	70	437	7.56 %
comp07	Ing0607-2	20	25	131	434	86.80 %	41.71 %	77	508	5.97 %
comp08	Ing0607-3	18	25	86	324	72.00 %	37.39 %	61	214	5.85 %
comp09	Ing0304-3	18	25	76	279	62.00 %	32.67 %	75	251	8.81 %
comp10	Ing0405-2	18	25	115	370	82.22 %	36.38 %	67	481	7.34 %
comp11	Fis0506-2	5	45	30	162	72.00 %	56.23 %	13	75	17.24 %
comp12	Let0506-2	11	36	88	218	55.05 %	35.06 %	150	1181	30.85 %
comp13	Ing0506-3	19	25	82	308	64.84 %	38.14 %	66	216	6.50 %
comp14	Ing0708-1	17	25	85	275	64.71 %	34.78 %	60	368	10.31 %

Table 2: Linear programming relaxations of a compact formulation of the Udine Course Timetabling problem, referred to as **Monolithic** by Burke et al. [2010], and of the subset of the proposed formulation with mildly exponential number of constraints we use at the root node, with all implied bounds added statically: dimensions of matrices after all automatic reductions in-built in CPLEX 10 and root relaxation time using default settings of CPLEX 10 (Dual Simplex) and manually tuned CPLEX 10 Barrier LP solver.

	Monolithic formulation		Proposed formulation		
Instance	Reduced LP dimensions	Barrier LP solver runtime at root node	Reduced LP dimensions at root node	Dual Simplex solver runtime at root node	Barrier LP solver runtime at root node
comp01	6484×5760	18.91 s	6516×5500	27.70 s	3.58 s
comp02	30034×27693	87.58 s	30128×26703	1185.02 s	54.90 s
comp03	26862×25489	80.91 s	26941×24563	674.44 s	49.97 s
comp04	33608×31265	63.83 s	33698×30525	1191.92 s	41.00 s
comp05	16189×14859	77.45 s	16259×12129	149.74 s	84.64 s
comp06	44050×41120	186.71 s	44168×40113	1798.53 s	73.17 s
comp07	60611×57012	510.05 s	60745×55895	1798.75 s	192.35 s
comp08	35642×33180	71.06 s	35735×32397	1175.29 s	43.25 s
comp09	32308×29979	88.31 s	32391×29024	1085.17 s	48.23 s
comp10	45870×43257	246.26 s	45996×42279	1798.97 s	105.03 s
comp11	8702×7126	9.31 s	8733×6672	26.03 s	7.69 s
comp12	27552×25007	295.08 s	27652×22117	669.02 s	134.76 s
comp13	35597×33200	70.13 s	35691×32353	1176.16 s	37.34 s
comp14	33288×30872	83.65 s	33384×30057	987.88 s	55.86 s

in the competition. (See Table 2.) Optimal solutions of yet larger real-life instances, such as those from Purdue [Rudová and Murray, 2003, Murray et al., 2007], seem to be completely out of reach for exact methods.

3 The Integer Programming Formulation

In order to model the Udine Course Timetabling problem using integer programming, it is necessary to choose decision variables. We view the problem as a variation of the three-index assignment [Balas and Saltzman, 1989] with side constraints. There is the temptation to use a binary variable for each event-room-period combination. This encoding harks back to Vlach [1967] and corresponds to the trivial formulation of graph colouring,

where the number of binary variables is the product of the number of vertices and an upper bound on the number of colours. There are, however, many alternative formulations of graph colouring [Burke et al., 2007] and it seems reasonable, instead of just using the trivial formulation, to use the formulation of graph colouring that would perform the best, considering the soft constraints. After exploring a number of such alternatives, Burke et al. [2007] proposed a formulation based on a clique-partition, which is given implicitly in many graph colouring applications. For Udine Course Timetabling, this formulation translates to a smaller number of binary decision variables, given by the product of the numbers periods, rooms, and courses, rather than events. This encoding will be used throughout this paper in the "core" decision variables x.

There are also four dependent variables v, y, w, and z, whose values are derived from the values of x in the process of solving. The variables and their meanings are:

- $x_{p,r,c}$ is one if course c is taught in room r at period p and zero otherwise
- $y_{r,c}$ is one if room r is used by course c and zero otherwise
- $v_{d,c}$ is one if there is at least one event of course c held on day d and zero otherwise
- w_c is the non-negative number of days course c is short of M^c, the recommended number of days of instruction
- $z_{u,d}$ is the penalty for undesirable patterns in the timetable of curriculum u for day d.

The constraints present in the initial relaxation force the value of $z_{u,d}$ to at least one if at least one pattern-penalising constraint is violated in the timetable for curriculum u and day d. Higher values of $z_{u,d}$ are enforced only dynamically, using Type 1 cuts described in Section 4

The objective function can be expressed as:

$$\min\ \mathrm{W}^{\mathrm{RCap}} \sum_{r\in\mathrm{R}}\sum_{p\in\mathrm{P}}\sum_{\substack{c\in\mathrm{C}\\ \mathrm{S}^c>\mathrm{A}^r}} x_{p,r,c}\,(\mathrm{S}^c-\mathrm{A}^r) + \mathrm{W}^{\mathrm{TCom}}\sum_{u\in\mathrm{U}}\sum_{d\in\mathrm{D}} z_{u,d}$$
$$+\ \mathrm{W}^{\mathrm{TSpr}}\sum_{c\in\mathrm{C}} w_c + \mathrm{W}^{\mathrm{RStb}}\sum_{c\in\mathrm{C}}\left(\left(\sum_{r\in\mathrm{R}} y_{r,c}\right)-1\right)$$

Hard constraints can be formulated as follows:

$$\forall c \in \mathrm{C} \qquad \sum_{p \in \mathrm{P}} \sum_{r \in \mathrm{R}} x_{p,r,c} = \mathrm{E}^c \qquad (1)$$

$$\forall p \in \mathrm{P}\, \forall r \in \mathrm{R} \qquad \sum_{c \in \mathrm{C}} x_{p,r,c} \leq 1 \qquad (2)$$

$$\forall p \in \mathrm{P}\, \forall c \in \mathrm{C} \qquad \sum_{r \in \mathrm{R}} x_{p,r,c} \leq 1 \qquad (3)$$

$$\forall p \in \mathrm{P}\, \forall t \in \mathrm{T} \qquad \sum_{r \in \mathrm{R}} \sum_{c \in \tilde{T}^t} x_{p,r,c} \leq 1 \qquad (4)$$

$$\forall p \in \mathrm{P}\, \forall u \in \mathrm{U} \qquad \sum_{r \in \mathrm{R}} \sum_{c \in \tilde{U}^u} x_{p,r,c} \leq 1 \qquad (5)$$

$$\forall \langle c, p \rangle \in \mathrm{N} \qquad \sum_{r \in \mathrm{R}} x_{p,r,c} = 0 \qquad (6)$$

Constraint (1) enforces a given number of events to be taught for each course. Constraints (3–5) stipulate that only one event within a single course or curriculum or taught by a single teacher can be held at any given period. Notice the similarity of constraints (1–5) and the supernodal formulation of graph colouring. Notice also that constraint (5) renders constraint (3) redundant, if there are no courses outside of any curricula. Finally, constraint (6) forbids the use of some periods in timetables of some courses, corresponding to a pre-colouring extension.

The formulation of soft constraints is less trivial. Values of x can be aggregated into v using constraints (7–8), in effect constructing daily timetables for individual curricula. Subsequently, the number of distinct weekdays of instruction course c is short of the prescribed value M^c, can be forced into w_c using constraint (9):

$$\forall c \in \mathrm{C}\, \forall d \in \mathrm{D}\, \forall p \in \tilde{D}^d \qquad \sum_{r \in \mathrm{R}} x_{p,r,c} \leq v_{d,c} \qquad (7)$$

$$\forall c \in \mathrm{C}\, \forall d \in \mathrm{D} \qquad \sum_{r \in \mathrm{R}} \sum_{p \in \tilde{D}^d} x_{p,r,c} \geq v_{d,c} \qquad (8)$$

$$\forall c \in \mathrm{C} \qquad \sum_{d \in \mathrm{D}} v_{d,c} \geq \mathrm{M}^c - w_c \qquad (9)$$

The "natural" formulation of penalisation of patterns [Burke et al., 2008] occurring in daily timetables of individual curricula goes through the daily timetables bit by bit, first checking isolated events in the first and the last period of the day, and later looking for triples of consecutive periods with only the middle period occupied by an event: For an instance with four

periods per day, this is:

$$\forall u \in \mathrm{U}\, \forall d \in \mathrm{D}\, \forall \langle p_1, p_2, p_3, p_4 \rangle \in \tilde{D}^d$$

$$\sum_{c \in \tilde{U}^u} \sum_{r \in \mathrm{R}} (x_{p_1,r,c} - x_{p_2,r,c}) \leq z_{u,d} \tag{10}$$

$$\sum_{c \in \tilde{U}^u} \sum_{r \in \mathrm{R}} (x_{p_4,r,c} - x_{p_3,r,c}) \leq z_{u,d} \tag{11}$$

$$\sum_{c \in \tilde{U}^u} \sum_{r \in \mathrm{R}} (x_{p_2,r,c} - x_{p_1,r,c} - x_{p_3,r,c}) \leq z_{u,d} \tag{12}$$

$$\sum_{c \in \tilde{U}^u} \sum_{r \in \mathrm{R}} (x_{p_3,r,c} - x_{p_2,r,c} - x_{p_4,r,c}) \leq z_{u,d} \tag{13}$$

Finally, values of y are constrained analogously to values of v:

$$\forall p \in \mathrm{P}\, \forall r \in \mathrm{R}\, \forall c \in \mathrm{C} \qquad x_{p,r,c} \leq y_{r,c} \tag{14}$$

These constraints complete the proposed formulation.

4 Cuts

This proposed formulation of Udine Course Timetabling has to use cuts from event/free-period patterns to be exact. As usual, we refer to these "necessary" cuts as "Type 1" cuts. We further describe two more classes of problem-specific cuts, which are not strictly necessary to reach optimality, but improve the performance considerably, when added on violation. We refer to those as "Type 2" cuts.

4.1 Cuts from Event/Free-Period Patterns (Type 1)

Constraints (10–13) penalising patterns of events and free periods in distinct daily timetables are not complete: The penalty $z_{u,d}$ incurred by each curriculum-day pair can be arbitrarily high, if the number of periods per day is a part of the input, although Constraints (10–13) cannot force it to any value higher than one. This forcing, however, can be done by enumeration of daily event/free-period patterns and penalties they incur [Burke et al., 2008]. When n is the number of periods per day, patterns in daily timetables of individual curricula can also be thought of as $B = \langle b_1, \ldots, b_n \rangle$, where b_i is equal to one if there is an event in the timetable of the given curriculum in period i of the given day and -1 otherwise. The number of ones or events in pattern B is denoted $m = \sum_{b \in B, b=1} b$. Each pattern is also associated with penalty p, which is the number of isolated events in the International Timetabling Competition 2007. If pattern B is to be penalised

with penalty p and constant m, we can formulate the constraint as:

$$\forall u \in \mathrm{U}\ \forall d \in \mathrm{D}\ \forall \langle p_1, p_2, \ldots, p_n \rangle \in \tilde{D}^d \quad p\left(1 - m + \sum_{i=1}^{n}\left(b_i \sum_{c \in \tilde{U}^u} \sum_{r \in \mathrm{R}} x_{p_i,r,c}\right)\right) \leq z_{u,d} \quad (15)$$

Notice that this constraint is linear.

For example penalisation of the pattern $B = \langle b_1, b_2, b_3 \rangle$ with penalty p and constant m in the case of three periods per day, the left hand side of the expression above (15) is:

$$p\left(1 - m + b_1 \sum_{c \in \tilde{U}^u} \sum_{r \in \mathrm{R}} x_{p_1,r,c} + b_2 \sum_{c \in \tilde{U}^u} \sum_{r \in \mathrm{R}} x_{p_2,r,c} + b_3 \sum_{c \in \tilde{U}^u} \sum_{r \in \mathrm{R}} x_{p_3,r,c}\right) \quad (16)$$

Out of the nine possible patterns for three periods per day, there is penalty 1 and constant 1 associated with $\langle -1, -1, 1 \rangle$ and $\langle 1, -1, -1 \rangle$ and penalty 2 and constant 2 associated with $\langle 1, -1, 1 \rangle$ in the International Timetabling Competition 2007. Let us assume there are events taught only in the first and last period of day d for curriculum u, corresponding to the patterns $\langle 1, -1, 1 \rangle$. The expression above (16) for pattern $\langle 1, -1, 1 \rangle$ evaluates to $2 \cdot (1 - 2 + 1 \cdot 1 - 1 \cdot 0 + 1 \cdot 1) = 2$, and we hence have $z_{u,d} \geq 2$. Let us now take a different example, where there is only one event taught in the first period of day d for curriculum u. The expression above (16) for pattern $\langle 1, -1, 1 \rangle$ evaluates to $2 \cdot (1 - 2 + 1 \cdot 1 - 1 \cdot 0 + 1 \cdot 0)$, which does not affect $z_{u,d}$ with lower bound of zero already. Nevertheless, for pattern $\langle 1, -1, -1 \rangle$ with penalty 1, the expression above (16) evaluates to $1 \cdot (1 - 1 + 1 \cdot 1 - 1 \cdot 0 - 1 \cdot 0) = 1$, and hence $z_{u,d} \geq 1$. Notice that the constraint with -1 and 1 coefficients cannot be obtained as a linear combination of the constraints penalising "feature by feature" (10–13) [Burke et al., 2008].

Burke et al. [2008] have shown that, although these constraints tend to be dense, the progress of the integer programming solver can often be sped up even by including all the constraints in the initial linear programming relaxation, solving the extended formulation statically, in effect. This is rather surprising, as the number of such constraints is mildly exponential. That is: the number of constraints is linear in the number of patterns, but the number of patterns is exponential in the number of periods per day. Hence, it seems more appropriate to add the constraints dynamically, only when they are violated. When there are no curricula u with $z_{u,d} \geq 2$ on any day d in the solution, as it is often the case, one needs only a modest number of cuts. In practice, adding the cuts from enumeration of event/free-period patterns only dynamically does indeed improve the performance considerably.

4.2 Implied Bounds and Cuts (Type 2)

Another class of problem-specific cuts and bounds can be seen as "implications" of the formulation presented in Section 3. For example, constraint (1) stipulates there have to be at least E^c events of course c in the weekly timetable. Constraints (7–8) set $v_{d,c}$ to one, if and only if there is an event of course c in the timetable for day d. Logically, it follows that for each course c with $\mathrm{E}^c > 0$, at least one of $v_{d,c}$ has to be set to one. However, even modern general integer programming solvers benefit from:

$$\forall c \in \mathrm{C} \qquad \sum_{d \in \mathrm{D}} v_{d,c} \geq 1 \tag{17}$$

$$\forall c \in \mathrm{C} \qquad \sum_{d \in \mathrm{D}} v_{d,c} \leq \mathrm{E}^c \tag{18}$$

Similarly, events of each course c take place at least in one room, but in fewer rooms than E^c:

$$\forall c \in \mathrm{C} \qquad \sum_{r \in \mathrm{R}} y_{r,c} \geq 1 \tag{19}$$

$$\forall c \in \mathrm{C} \qquad \sum_{r \in \mathrm{R}} y_{r,c} \leq \mathrm{E}^c \tag{20}$$

In a similar fashion, we can add also:

$$\forall r \in \mathrm{R}\, \forall c \in \mathrm{C} \qquad \sum_{p \in \mathrm{P}} x_{p,r,c} \geq y_{r,c} \tag{21}$$

$$\forall c \in \mathrm{C} \qquad w_c \leq \mathrm{M}^c - 1 \tag{22}$$

Finally, we can add:

$$\forall r \in \mathrm{R}\, \forall c \in \mathrm{C} \qquad \sum_{p \in \mathrm{P}} x_{p,r,c} \leq \mathrm{E}^c y_{r,c} \tag{23}$$

Perhaps surprisingly, these simple constraints improve the performance of modern integer programming solvers by an order of magnitude, even when added statically. It might hence be interesting to study the automation of their generation for general integer programming.

4.3 Cuts from Numbers of Days of Instruction (Type 2)

Another large class of cuts comes from the soft constraints penalising insufficient numbers of distinct numbers of days of instruction per course. The soft constraint implies the number of events of course c taking place on a single day cannot be higher than one plus the number of events not necessary to maintain the spread of events throughout the week ($\mathrm{E}^c - \mathrm{M}^c$),

if penalty w_c is to be zero for course c. If we take into account the number of violations of the correponding soft constraint (w_c), we obtain:

$$\forall c \in \mathrm{C}\, \forall d \in \mathrm{D} \qquad \sum_{p \in \tilde{D}^d} \sum_{r \in \mathrm{R}} x_{p,r,c} \leq 1 + \mathrm{E}^c - \mathrm{M}^c + w_c \quad (24)$$

This constraint then can be naturally extended to cover two days:

$$\forall c \in \mathrm{C}\, \forall \{d_1, d_2\} \subset \mathrm{D}, d_1 \neq d_2 \quad \sum_{p \in \tilde{D}^{d_1} \cup \tilde{D}^{d_2}} \sum_{r \in \mathrm{R}} x_{p,r,c} \leq 2 + \mathrm{E}^c - \mathrm{M}^c + w_c \quad (25)$$

where $d > d_1$ uses the usual order of days within a week. In general, it is possible to come up with constraints linking an arbitrary number n of days:

$$\forall c \in \mathrm{C}\, \forall d_1 \in \mathrm{D}\, \forall d_2 \in \{d_2 \in \mathrm{D} \mid d_2 > d_1\} \ldots \forall d_n \in \{d_n \in \mathrm{D} \mid d_n > d_{n-1}\} \quad \sum_{p \in \bigcup_{i=1}^{n} \tilde{D}^{d_i}} \sum_{r \in \mathrm{R}} x_{p,r,c} \leq n + \mathrm{E}^c - \mathrm{M}^c + w_c \quad (26)$$

One could also combine these constraints with constraints penalising event/free-period patterns. It seems, however, that both extensions are not violated often enough to merit the separation and addition of such dense constraints.

4.4 Cuts from Graph Colouring (Type 2)

Several large classes of cuts can also be obtained from the bounded graph colouring component implicit in Udine Course Timetabling. Although the usual linear programming relaxations are known to be rather weak [Caprara, 1998], there are a number of known classes of cuts [Coll et al., 2002, Campêlo et al., 2003]. In timetabling terms, the strongest class, the clique cuts, corresponds to:

$$\forall q \in \mathrm{Q}\, \forall p \in \mathrm{P} \qquad \sum_{r \in \mathrm{R}} \sum_{c \in q} x_{p,r,c} \leq 1, \quad (27)$$

where Q is a collection of subsets of courses, which correspond to complete subgraphs in the course-based conflict graph. The strength of these cuts is known both in theory and practice: Chvátal [1973] has shown that for perfect graphs, the relaxation with cuts from cliques is a convex hull of integer points; Méndez-Díaz and Zabala [2008] compare clique cuts to other known classes of cuts empirically; finally, as Table 3 suggests, for timetabling instances with dense enough conflict graphs, clique cuts improve the performance considerably.

4.5 Cuts Exploiting Integrality of the Objective (Type 2)

Finally, and perhaps most interestingly to the wider optimisation community, we add cuts exploiting the integrality of the objective. Unfortunately, even when all variables in the objective are integral, CPLEX[1] does not round up a fractional lower bound. We hence add this feature. We add both local cuts based on the current lower bound, and global cuts based on the least lower bound among the active nodes, whenever their respective fractional part is in the interval between 0.01 and 0.99. The check for the interval is meant to ensure we do not, accidentally, round up an imprecise floating point representation of a whole number. This, again, seems to reduce the run-time to optimality considerably. For example on instance comp01 from the International Timetabling Competition, for which there is an easily obtainable lower bound of 4 and optimum of 5, the reduction of run-time is by manyfold.

5 The Implementation

A solver based on the cuts presented above has been implemented in C++ using IBM/ILOG Concert libraries and using numerous routines from IBM/ILOG CPLEX [ILOG, 2006], the integer programming solver. The solver has been tested on Microsoft Windows, Linux, and Oracle Solaris. Its source code is freely available on-line[2] under GNU General Public License.

More specifically, we have used in-built presolve, linear programming, Chvátal-Gomory and related general-purpose cut separation, strong branching, and search tree management routines of CPLEX in versions 10 and 12.1. Our implementation added separation routines for cuts from event/free-period patterns, cuts from numbers of days of instruction, cuts exploiting integrality of the objective, and cuts from cliques. It applies implied bounds and constraints statically, starting from root-node. As an extension, we have also implemented a "lower bounding heuristic", based on an alternative enforcement of integrality.

5.1 Separation Routines

Cut separation routines check for any violation of the cuts described above, and add them into the clique pool of CPLEX, if violated. The separation routines are run at every node at least once, and repeatedly at the root node. We do not restrict the number of cuts added at any node and do not check the "numerical safety" of the cuts in any fashion. We let CPLEX to include cuts from the clique pool in linear programming relaxations and to remove them, when they are no longer tight.

[1] IBM/ILOG CPLEX Version 12.1, current at the time of writing (June 2010).
[2] http://cs.nott.ac.uk/~jxm/timetabling/bc/ (June 2010)

Our approach to separation of cuts from event/free-period patterns can best be described as "brute force". For each curriculum and each daily timetable of the curriculum, we go through all patterns to check for violations. As the number of patterns is exponential in the number of periods per day, we have not been able to show a polynomial worst-case upper bound on the runtime of the separation routine. Due to the low number of periods per day in the instances we have encountered, however, it seems to perform well. Similarly, the number of days of instruction patterns is exponential in the number of days in a week. It seems reasonable, however, to consider the number of days in a week a constant, and hence a "brute force" check for violations of inequalities from numbers of days of instruction runs in polynomial time.

For clique cuts, we are using two separation routines. In the first routine, we check for violations of all maximal cliques larger than three vertices, which are stored in a clique pool populated at the root node using a branch and bound code of Niskanen and Östergøard [2003], within the reported time limit. Although the number of maximal cliques in the graph is exponential in the number of vertices, in theory, instances from the International Timetabling Competition 2007 tend to have only a small number of large maximal cliques. In the second routine, we check for violations of triangles. Previously, we have also attempted with a routine that tried to "grow" violated triangle inequalities into larger cliques. This, however, does not necessarily improve the performance, as it introduces a number of non-maximal cliques. In both clique cut separation routines, we benefit from using a course-based conflict graph [Burke et al., 2007], which is much smaller than the event-based one.

5.2 Another Relaxation

As an extension, we have also implemented a "lower bounding heuristic", based on an alternative enforcement of integrality. Such a heuristic is in some sense dual to primal heuristics, which are present in all modern integer programming solvers. Similarly to modern primal heuristics, it is based on the "submip" model, where we solve a smaller instance of integer linear programming to obtain the bound. In particular, we relax the integrality of the core decision variables x, while enforcing the integrality of aggregate course-day (v), course-room (y), and pattern (z) variables. Often, this results in an order of magnitude smaller number of binary variables, allowing for a very fast bounding procedure. For example on the instance comp01 from the International Timetabling Competition 2007, the lower bound thus obtained is the best possible. In general, this procedure is a relaxation stronger than the ordinary linear programming relaxation. In certain special cases, where the penalties are low, solutions of the relaxation may overlap with the solutions to the full problem.

5.3 Settings and Tuning

A number of CPLEX settings need to be changed from their defaults in order to ensure correct behaviour. Most notably, primal heuristics solving another instance of integer programming, which are not aware of Type 1 cuts required, tend not to produce usable solutions. To this end, we set parameters HeurFreq, RINSHeur, and FPHeur all to -1. Caution needs to be taken also in warm-starting the solver. Solutions produced by other types of primal heuristics need to be checked for the correctness of their objective function value.

A limited experimentation with "manual" tuning of CPLEX settings has been attempted on the instances from the International Timetabling Competition 2007. The default auto-tuning procedures built in CPLEX are often unable to pick the most appropriate method of linear programming (LP). Hence, we have implemented an auto-tuning procedure that solves a much smaller instance of linear programming, effectively implementing the bounded graph colouring component, using various LP solvers. (This formulation is denoted as **Surface** in the study of Burke et al. [2010] and its LP relaxation can be often solved within a single second for instances from the International Timetabling Competition 2007.) We then use the LP solver performing best on the much smaller formulation for solving the larger instance. Experiments confirm that this choice corresponds to the best performing solver on the formulation presented in this paper on the set of 14 instances. Running all linear programming solvers concurrently, however, achieves the same result at the cost of a single parameter switch (RootAlg = 6) and no change in wall-clock run-time on a multi-core machine.

6 Computational Experience

The implementation has been evaluated using the Linux version of CPLEX version 10.00 restricted to run on a single processor of a desktop PC equipped with two Intel Pentium 4 processors clocked at 3.20 GHz and with 2 GB of RAM. All computations were restricted to a single processor. The timetabling benchmark[3] by Bonutti et al. [2011] runs for 780 seconds when no other computationally demanding processes are running on the computer. When, for comparison, we provide lower bounds obtained by Lach and Lübbecke [2011] and upper bounds obtained the winning solver of Track 3 in the International Timetabling Competition (ITC) 2007, developed by Müller [2008], we normalise their run times using this benchmark.

The performance of the proposed solver varies widely from instance to instance. For relatively easier instances comp01 and comp11 from the ITC 2007, root relaxations obtained using the proposed formulation take 3.58 s and 7.69 s, respectively, to solve using ILOG CPLEX 10 Barrier LP Solver. In

[3] http://www.cs.qub.ac.uk/itc2007/benchmarking/ (June 2010)

Table 3: Lower bounds obtained using various formulations: The proposed formulation uses cuts from event/free-period patterns and implied bounds ("IB") in all runs, with optional addition of cuts from days of instruction patterns (denoted "patterns") or clique-cuts ("cliques"). Further results are forthcoming.

	Lower bounds							Upper b.
Instance	Burke et al. [2010]: **Monolithic** (30 min. w/o cuts)	Proposed formulation (30 min. w/ IB)	Burke et al. [2010]: **Surface** (30 min. w/ IB)	Proposed formulation (30 min. w/ IB and cliques)	Proposed formulation (30 min. w/ IB and patterns)	Lach and Lübbecke [2011] (root w/ CPLEX 11)	Lach and Lübbecke [2011] (58.5 min. w/ CPLEX 11)	Müller [2008] (10 times 12.6 min.)
comp01	-11	4	5	4	5	4	4	5
comp02	-59	0	0	0	6	0	8	51
comp03	-53	0	2	0	43	1	23	84
comp04	-56	0	0	0	2	12	27	37
comp05	-29	17	33	26	183	93	101	330
comp06	-71	6	6	6	6	7	7	48
comp07	-98	0	0	0	0	0	0	20
comp08	-56	0	1	0	2	2	34	41
comp09	-57	0	6	0	0	19	40	109
comp10	-86	0	0	0	0	2	4	16
comp11	-17	0	0	0	0	0	0	0
comp12	-58	7	12	4	5	31	32	333
comp13	-56	0	4	0	0	20	37	66
comp14	-62	0	23	0	0	40	41	59

the remaining instances, the root relaxation is yet more expensive, taking up to 192 seconds on comp07. (It should be noted that ILOG CPLEX 10 using the default settings or auto-tuning chooses CPLEX Dual Simplex LP Solver, which results in dramatically higher LP solver run times.) Lower bounds obtained from the LP relaxation at the root node do not seem to suffer from the need to add Type 1 cuts to achieve optimality, and thanks to the static addition of cuts from implied bounds provide considerably better lower bounds than the monolithic formulation of Burke et al. [2007]. For instance for comp05, it is possible to obtain lower bound of 33 after 48 hours, by tuning the solver for "best bound" and using cuts from implied bounds. When we apply also the cuts from patterns, it is possible to obtain

the lower bound of 183 within 30 minutes. For complete results obtained using ILOG CPLEX 10 Barrier LP Solver, see Table 2.

Together, the good lower bounds and the low run time obtained using the Barrier LP solver, seem to suggest that the integer programming solver should be able to make progress fast. Indeed, provably optimal solutions for instances comp01 and comp11 can be obtained within fifteen minutes of run time. Results of longer runs are forthcoming and will be made available in a later version of this paper.

7 Related Work

The related work stems from the properties of the Udine Course Timetabling polytope, suggested in Section 3. The polytope is not full-dimensional and is related to the three-index assignment polytope, the bounded graph colouring polytope, and the the weighted matching polytope, albeit the first two are not plain projections.

The matching polytopes, including the general matching and perfect matching polytopes, are among the best studied in polyhedral combinatorics. The matching polytope of G is the convex hull of incidence vectors x of matchings in G. Edmonds [1965] has show that for any $G = (V, E)$, the matching polytope is described completely by:

- non-negativity constraints, $x_e \geq 0$ for each $e \in E$,
- matching constraints $\sum_{e \in \delta(v)} x_e \leq 1$ for each $v \in V$,
- cuts from odd subsets, $\sum_{e \in E(U)} x_e \leq \lfloor |U|/2 \rfloor$ for each $U \subseteq V$ with $|U|$ odd.

Further, for bipartite G, one does not need the cuts from odd subsets. Our understanding of the polytopes related to NP-Hard problems in much less complete.

There is a long history to the study of three-index assignment problems Vlach [1967]. Generally, given three disjoint sets I, J, K and weights $c_{i,j,k}$ associated with triplets $(i, j, k) \in I \times J \times K$, one has to find a minimum-weight collection of "disjoint" triplets, where by "disjoint" one understands disjoint with respect to all three pairs of indices. There are two variants in use:

- in the *axial* problem, the goal is to find n triplets based on the given sets of equal cardinality $|I| = |J| = |K|$
- in the *planar* problem, the goal is to find n disjoint sets of n disjoint triplets each.

The axial three-index assignment polytope has been studied by Balas and Saltzman [1989], Balas and Saltzman [1991], Gwan and Qi [1992], Balas and Qi [1993], and others. The most important cuts for the axial three-index assignment problem are:

- cuts from cliques

- cuts from odd holes
- cuts from combs
- cuts from bulls.

Although the problem itself is rather different from Udine Course Timetabling, it seem the cuts exploit structures that are common to the two problems.

The Udine Course Timetabling Polytope is closely related to the Bounded Graph Colouring Polytope. The graph colouring polytopes have recently been studied by Coll et al. [2002], Campêlo et al. [2003] and Méndez-Díaz and Zabala [2008]. The most important cuts for the axial three-index assignment problem are:

- cuts from cliques
- cuts from odd holes
- cuts from odd antiholes
- cuts from paths
- symmetry-breaking block colour inequalities
- neighbourhood inequalities replacing edgewise inequalities
- multicolour inequalities strengthening edgewise inequalities.

Unfortunately, Méndez-Díaz and Zabala [2008] seem to have conceded that no combination of these cuts is clearly superior to clique cuts alone, when the run time of separation routines is taken into account. Avella and Vasil'ev [2005], however, seem to suggest that cuts from odd holes are helpful in branch and cut routines for timetabling, although they do not provide much evidence. Performance of the remaining cuts on instances from timetabling is yet to be investigated.

8 Conclusions

We have proposed a branch-and-cut procedure for Udine Course Timetabling, a university course timetabling problem with multiple soft constraints.

The procedure uses a formulation, which reduces the number of variables necessary to formulate the soft constraints, at the cost of working with a mildly exponential number of constraints, which can be added upon violation. The gains obtained from the change of formulation in the terms of linear-programming solver run-time are bounded from above by factor of 10, when the linear programming solver is auto-tuned, and by two or three orders of magnitude, when the default linear programming solver settings inbuilt in ILOG CPLEX 10 are used. The runtime of the linear programming solver, however, might still hinder the applicability of this approach to considerably larger instances [Rudová and Murray, 2003, Murray et al., 2007].

Several large problem-specific classes of valid inequalities have been proposed. Cuts exploiting integrality of the objective function value, in particular, seem to have wider applicability. Together, the cuts do improve

the performance considerably. An integer programming relaxation, which enforces integrality only in certain aggregate variables, could also be of independent interest.

For two instances of Udine Course Timetabling used in the International Timetabling Competition 2007, the procedure provides the best possible lower bounds with fifteen minutes. Further optima are arguably obtainable within longer run-times. Some of the concepts seem to be easy to apply to related timetabling problems with soft constraints.

Future work could focus on exploring further cuts from graph colouring, three-index assignment, and weighted matching, exploring alternative branching rules, and primal and improvement heuristics targeted specifically at timetabling applications.

Acknowledgments The authors are grateful to Andrea Schaerf and Luca Di Gaspero, who kindly maintain the Udine Course Timetabling Problem, including all data sets, and to the referees for their careful reading of the paper and stimulating questions. Andrew Parkes has been supported by the UK Engineering and Physical Sciences Research Council under grant GR/T26115/01. Hana Rudová has been supported by project MSM0021622419 of Ministerstvo školství, mládeže a tělovýchovy.

References

Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network flows: Theory, algorithms, and applications*. Prentice-Hall, 1993.

Salem Mohammed Al-Yakoob and Hanif D. Sherali. A mixed-integer programming approach to a class timetabling problem: A case study with gender policies and traffic considerations. *European J. Oper. Res.*, 180(3): 1028–1044, 2007.

Pasquale Avella and Igor Vasil'ev. A computational study of a cutting plane algorithm for university course timetabling. *J. Scheduling*, 8(6):497–514, 2005.

Egon Balas and Alexander Bockmayr and Nicolai Pisaruk, and Laurence Wolsey On unions and dominants of polytopes. *Math. Program.*, 99(2): 223–239, 2004.

Egon Balas and Li Qun Qi. Linear-time separation algorithms for the three-index assignment polytope. *Discrete Appl. Math.*, 43(1):1–12, 1993.

Egon Balas and Matthew J. Saltzman. Facets of the Three-Index Assignment Polytope. *Discrete Appl. Math.*, 23: 201–229, 1989.

Egon Balas and Matthew J. Saltzman. An algorithm for the three-index assignment problem. *Oper. Res.*, 39(1):150–161, 1991.

Victor A. Bardadym. Computer-aided school and university timetabling: The new wave. In Edmund K. Burke and Peter Ross, editors, *Practice and Theory of Automated Timetabling*, volume LNCS 1153 of *Lecture Notes in Computer Science*, pages 22–45, Berlin, 1996. Springer.

Camille B. Beyrouthy, Edmund K. Burke, Dario Landa Silva, Barry McCollum, Peter McMullan, and Andrew J. Parkes. Towards improving the utilisation of university teaching space. *J. Op. Res. Soc.*, 60(1): 130–143, 2009.

Alex Bonutti, Fabio De Cesco, Luca Di Gaspero, and Andrea Schaerf. Benchmarking curriculum-based course timetabling: formulations, data formats, instances,

validation, visualization, and results. *Ann. Oper. Res.*, this volume, this year. doi: 10.1007/s10479-010-0707-0

Edmund K. Burke and Sanja Petrovic. Recent research directions in automated timetabling. *European J. Oper. Res.*, 140(2):266–280, 2002.

Edmund K. Burke, Kirk Jackson, Jeffrey H. Kingston, and Rupert F. Weare. Automated university timetabling: The state of the art. *Comput. J.*, 40(9):565–571, 1997.

Edmund K. Burke, Dominique de Werra, and Jeffrey H. Kingston. Applications to timetabling. In Jonathan L. Gross and Jay Yellen, editors, *Handbook of Graph Theory*, pages 445–474. CRC, London, UK, 2004.

Edmund K. Burke, Jakub Mareček, Andrew J. Parkes, and Hana Rudová. A supernodal formulation of vertex colouring with applications in course timetabling. *Ann. Oper. Res*, to appear. doi: 10.1007/s10479-010-0716-z

Edmund K. Burke, Jakub Mareček, Andrew J. Parkes, and Hana Rudová. Penalising patterns in timetables: Novel integer programming formulations. In Stefan Nickel and Jörg Kalcsics, editors, *Operations Research Proceedings 2007*, pages 409–414. Springer, Berlin, 2008.

Edmund K. Burke, Jakub Mareček, Andrew J. Parkes, and Hana Rudová. Decomposition, reformulation, and diving in timetabling. *Comput. Oper. Res.*, 37(1): 582–597, 2010.

Manoel Campêlo, Ricardo C. Corrêa, and Yuri Frota. Cliques, holes and the vertex coloring polytope. *Inform. Process. Lett.*, 89(4):159–164, 2003.

Alberto Caprara. Properties of some ILP formulations of a class of partitioning problems. *Discrete Appl. Math.*, 87(1-3):11–23, 1998.

Michael W. Carter. A Lagrangian relaxation approach to the classroom assignment problem. *INFORMS J. Comput.*, 27:230–245, 1989.

Michael W. Carter and Gilbert Laporte. Recent developments in practical course timetabling. In Edmund K. Burke and Michael W. Carter, editors, *Practice and Theory of Automated Timetabling*, volume LNCS 1408 of *Lecture Notes in Computer Science*, pages 3–19, Berlin, 1997. Springer.

Vašek Chvátal. Edmonds polytopes and a hierarchy of combinatorial problems. *Discrete Math.*, 4:305–337, 1973.

Pablo Coll, Javier Marenco, Isabel Méndez-Díaz, and Paula Zabala. Facets of the graph coloring polytope. *Ann. Oper. Res.*, 116:79–90, 2002.

Artur Czumaj and Andrzej Lingas. Finding a heaviest triangle is not harder than matrix multiplication. In *SODA '07: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 986–994, Philadelphia, PA, 2007. SIAM.

Sophia Daskalaki, Theodore Birbas, and Efthymios Housos. An integer programming formulation for a case study in university timetabling. *European J. Oper. Res.*, 153:117–135, 2004.

Sophia Daskalaki, Theodore Birbas, and Efthymios Housos. Efficient solutions for a university timetabling problem through integer programming. *European J. Oper. Res.*, 160:106–120, 2005.

Maria Dimopoulou and Panagiotis Miliotis. An automated university course timetabling system developed in a distributed environment: A case study. *European J. Oper. Res.*, 153:136–147, 2004.

Edmonds, Jack. Maximum matching and a polyhedron with $0,1$-vertices. *J. Res. Nat. Bur. Standards Sect. B*, 69B: 125–130, 1965.

Jack Edmonds, László Lovász, and William R. Pulleyblank. Brick decompositions and the matching rank of graphs. *Combinatorica*, 2(3): 247–274, 1982.

Luca Di Gaspero and Andrea Schaerf. Multi neighborhood local search with application to the course timetabling problem. In Edmund K. Burke and Patrick De Causmaecker, editors, *Practice and Theory of Automated Timetabling*, pages 262–275. Volume LNCS 2740 of *Lecture Notes in Computer Science*, Berlin, 2003. Springer.

Luca Di Gaspero and Andrea Schaerf. Neighborhood portfolio approach for local search applied to timetabling problems. *J. Math. Model. Algorithms*, 5(1):65–89, 2006.

Geena Gwan and Li Qun Qi. On facets of the three-index assignment polytope. *Australas. J. Combin.*, 6:67–87, 1992.

ILOG. *ILOG CPLEX Advanced Programming Techniques*. ILOG S. A., Incline Village, NV, 2006.

Gerald Lach and Marco E. Lübbecke. Optimal university course timetables and the partial transversal polytope. In Catherine C. McGeoch, editor, *Experimental algorithms*, volume LNCS 5038 of *Lecture Notes in Computer Science*, pages 235–248, Berlin, 2008. Springer.

Gerald Lach and Marco E. Lübbecke. Curriculum based course timetabling: new solutions to Udine benchmark instances. *Ann. Oper. Res.*, this volume, this year. doi: 10.1007/s10479-010-0700-7

Norman L. Lawrie. An integer linear programming model of a school timetabling problem. *Comput. J.*, 12:307–316, 1969.

Barry McCollum. A perspective on bridging the gap between theory and practice in university timetabling. In Edmund K. Burke and Hana Rudová, editors, *Practice and Theory of Automated Timetabling*, volume LNCS 3867 of *Lecture Notes in Computer Science*, pages 3–23, Berlin, 2007. Springer.

Barry McCollum, Andrea Schaerf, Ben Paechter, Paul McMullan, Rhyd Lewis, Andrew J. Parkes, Luca Di Gaspero, Rong Qu, Edmund K. Burke. Setting the Research Agenda in Automated Timetabling: The Second International Timetabling Competition. *INFORMS J. on Comput.*, 22(1): 120–130, 2010.

Isabel Méndez-Díaz and Paula Zabala. A cutting plane algorithm for graph coloring. *Discrete Appl. Math.*, 156(2), 2008.

S. A. Mirhassani. A computational approach to enhancing course timetabling with integer programming. *Appl. Math. Comput.*, 175:814–822, 2006.

Tomáš Müller. ITC-2007 solver description: A hybrid approach. *Ann. Oper. Res*, 127(1): 429–446, 2009.

Sampo Niskanen and Patric R. J. Östergøard. Cliquer User's Guide, Version 1.0. Communications Laboratory, Helsinki University of Technology, Espoo, Finland, Tech. Rep. T48, 2003.

Sanja Petrovic and Edmund K. Burke. University timetabling. In Joseph Leung, editor, *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, pages 1001–1023. CRC Press, Boca Raton, FL, 2004. ISBN 1584883979.

Andrea Qualizza and Paolo Serafini. A column generation scheme for faculty timetabling. In Edmund K. Burke and Michael A. Trick, editors, *Practice and Theory of Automated Timetabling*, volume LNCS 3616 of *Lecture Notes in Computer Science*, pages 161–173, Berlin, 2004. Springer.

Hana Rudová and Keith Murray. University course timetabling with soft constraints. In Edmund K. Burke and Patrick De Causmaecker, editors, *Practice and Theory of Automated Timetabling*, pages 310–328. Volume LNCS 2740 of *Lecture*

Notes in Computer Science, Berlin, 2003. Springer.

Hana Rudová, Tomáš Müller, and Keith Murray. Complex university course timetabling. *J. Scheduling*, to appear. doi: 10.1007/s10951-010-0171-3

Andrea Schaerf. A survey of automated timetabling. *Artificial Intelligence Rev.*, 13 (2):87–127, 1999.

Katja Schimmelpfeng and Stefan Helber. Application of a real-world university-course timetabling model solved by integer programming. *OR Spectrum*, 29: 783–803, 2007.

Etsuji Tomita, Akira Tanaka, and Haruhisa Takahashi. The worst-case time complexity for generating all maximal cliques and computational experiments. *Theor. Comput. Sci.*, 363(1):28–42, 2006.

Arabinda Tripathy. School timetabling – A case in large binary integer linear programming. *Management Sci.*, 30:1473–1489, 1984.

Milan Vlach. Branch and bound method for the three-index assignment problem. *Ekonom.-Mat. Obzor*, 3:181–191, 1967.

Dominic J. A. Welsh, and Martin B. Powel. An upper bound for the chromatic number of a graph and its application to timetabling problems. *Computer J.*, 10 (1):85–86, 1967.

Paula Zabala and Isabel Méndez-Díaz. A branch-and-cut algorithm for graph coloring. *Discrete Appl. Math.*, 154(5):826–847, 2006.

David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. *Theory Comput.*, 3(6):103–128, 2007.