Les Cahiers du GERAD ISSN: 0711–2440

Daily course pattern formulation and valid inequalities for the curriculum-based course timetabling problem

N.-C. Fink Bagger, G. Desaulniers
J. Desrosiers

G–2016–71

September 2016

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

Daily course pattern formulation and valid inequalities for the curriculum-based course timetabling problem

Niels-Christian Fink Bagger [a,b]

Guy Desaulniers [c]

Jacques Desrosiers [d]

[a] *mORetime Research Group, Management Science, Department of Management Engineering, Technical University of Denmark, Produktionstorvet, Building 426, 2800 Kgs. Lyngby, Denmark*

[b] *MaCom A/S, Vesterbrogade 48, 1., DK-1620 København V, Denmark*

[c] *GERAD & Polytechnique Montréal, Montréal (Québec) Canada H3C 3A7*

[d] *GERAD & HEC Montréal, Montréal (Québec) Canada H3T 2A7*

nbag@dtu.dk
guy.desaulniers@gerad.ca
jacques.desrosiers@gerad.ca

September 2016

Les Cahiers du GERAD

G–2016–71

Abstract: In this paper, we propose an integer programming model for obtaining lower bounds for the curriculum-based course timetabling problem, in which weekly assignments for courses into rooms and periods are considered. The model is a pattern formulation where a pattern is an assignment of a course into a set of periods on one day. Different preprocessing techniques are implemented to reduce the number of variables and valid inequalities are derived and added to the model. The proposed model is tested on 21 real-world data instances. On 17 of these instances the best know solutions have been proven optimal and for the remaining four our model improves the lower bounds for three of them.

Keywords: Course timetabling, integer programming, preprocessing, valid inequalities

Acknowledgments: Niels-Christian Fink Bagger is partly funded by the Innovation Fund Denmark (IFD).

1 Introduction

The problem considered in this paper is the *Curriculum-based Course Timetabling Problem* (CCT) as described by Di Gaspero et al (2007). The interest in this problem has increased as it was used as one of three problems for the *Second International Timetabling Competition* (ITC2007) in 2007 (Di Gaspero et al, 2007; McCollum et al, 2010). In connection to ITC2007, a website was developed (Bonutti et al, 2016) where it is possible for researchers to upload solutions and to report refined lower bounds. Twenty-one data sets were provided for the competition based on real world instances. Seven of these were hidden until after the competition. For the 21 data sets, most of them has been *closed*, meaning that the best solutions found have been proven optimal. However there are still four *open* data sets left and the aim of this paper is to reduce the gaps by improving the lower bounds.

1.1 Problem description

A set of courses is given, each having a predetermined number of lectures to be scheduled. Each course is taught be exactly one lecturer. A set of curricula is also given, each consisting of a set of courses. A course can belong to several curricula. Furthermore a time horizon in terms of a number of days that all the lectures need to be scheduled in is given. Each day is divided into a number of uniform time slots and a day and time slot pair is referred to as a period. It is assumed that each lecture takes up exactly one period. Besides scheduling the lectures into periods, it is also needed to choose a room for the lecture to take place in. Finally some constraints are also given. These can either be *hard* or *soft*. Any feasible timetable should fulfil all the hard constraints, i.e., a timetable is considered feasible if and only if all the hard-constraints have no violations. The objective is then to find a feasible timetable while minimizing a weighted sum of the violation of the soft constraints.

The problem formulated by Di Gaspero et al (2007) contains four hard constraints as listed below:

Availability (A): A course can have specific periods defined as *unavailable* periods. Every lecture scheduled in such a period is considered as one violation.

Conflicts (C): A pair of lectures is considered as conflicting if they belong to the same curriculum or are taught by the same lecturer. If a conflicting pair of lectures is scheduled in the same period, this is considered as one violation.

Lectures (L): Each course has a predetermined number of lectures to be scheduled in different periods. If a lecture is not scheduled or if two lectures of a course are scheduled in the same period, this is counted as one violation.

Room Occupancy (RO): For every room and every period, at most one lecture should be scheduled. For every room and period, every additional lecture is considered as one violation.

In our approach, we are only focusing on scheduling the courses into the periods. The only hard constraint involving the rooms is **RO** and this constraint can still be fulfilled since it is possible to put any course into any room. Therefore we just have to ensure that we do not schedule more courses into a single period than the number of available rooms in the data instance. The problem also contains four soft constraints for which we minimize the number of violations. These are listed below:

Isolated Lectures (IL): It is desired to create compact schedules. A lecture is *isolated* if no other lectures are scheduled for a curriculum on the same day in either a directly preceding or successive time slot (i.e., in consecutive time slots). Each isolated lecture is counted as one violation.

Example 1 *Consider a curriculum containing courses C001 and C002, a day with six time slots, and the following schedules on that day:*

C001:		*R001*				
C002:				*R002*		

The schedules shows that course C001 has a lecture in the second time slot while C002 has a lecture in the fourth time slot. Since these time slots are not adjacent, this means that the schedules count two violations on that day. Consider now the following schedules:

C001:				*R001*		
C002:			*R002*			

Course C001 has a lecture in the fourth time slot while C002 has a lecture in the third one. These time slots are adjacent and these schedules do not count any violations on that day.

Minimum Working Days (**MWD**)**:** For each course, a day is defined as a *working day* if at least one lecture is scheduled on it. For each course, a minimum number of working days is requested and each day below this minimum counts as one violation.

Room Capacity (**RC**)**:** There is a number of students attending a given course and each room has a specific capacity. If the number of students assigned is greater than the room capacity, then every student that cannot be accommodated counts as one violation.

Room Stability (**RStab**)**:** For every course, it is desired to schedule all lectures in just one room. Every additional room is counted as one violation.

As mentioned before, we only consider the scheduling of the courses into periods and ignore the assignment of rooms. Therefore we can only take the soft constraints **IL** and **MWD** into account which means that our approach is a relaxation of the original problem. Hence any valid lower bounds for our model are also valid lower bounds for the original problem.

It should be noted that the constraint **IL** is usually referred to as *curriculum compactness* in the literature. We have chosen to name it *isolated lectures* as multiple ways of formulating curriculum compactness are described in Bonutti et al (2012) and this is the name they give for this formulation.

1.2 Related work

As our focus is on lower bounds, we focus on the literature that have reported lower bounds as well. For the interested reader, we refer to Bettinelli et al (2015) as they give more detailed descriptions of all the methods presented here as well as descriptions of many more methods including those to obtain upper bounds.

Burke et al (2008, 2010, 2012) describes a compact Integer Linear Programming (ILP) formulation which can be solved by a commercial ILP solver. They discover that this formulation is very hard to solve and tackle this in various ways. In Burke et al (2010), one way to get a lower bound is to consider a relaxed version of the problem by setting the weights of the two room-related constraints **RC** and **RStab** to zero. This allows them to only consider the time-scheduling part of the problem. Furthermore a constraint is added to ensure that no more courses are scheduled in any time period than the total number of rooms available. Note that this is the same relaxation of the original problem that we are considering. They note that the before mentioned approach can be thought of as aggregating all the rooms into a single *multi-room*, where the capacity of this room is equal to the largest room. Another approach is to divide the rooms into sets of multi-rooms. For each multi-room, the capacity is equal to the largest room in the set and the maximum number of lectures that can take place in a multi-room in any period is equal to the number of rooms in the set. This allows them to partially include the room related soft constraints. After finding a solution to one of the before mentioned relaxations, the rooms are included in the model again and the solution obtained from the relaxation is used to guide the search by either fixing the periods that the lectures are assigned to or by fixing the days. In Burke et al (2008), they enumerate all patterns for each day and for each curriculum and add the costs of these patterns as cuts. These cuts are also used in Burke et al (2012) where they develop a cutting plane procedure and the cuts from Burke et al (2008) are added dynamically as they are violated. Furthermore other cuts are derived including *clique cuts* and some more problem specific cuts. The reported results show that the cutting plane approach leads to better performances compared to the compact ILP formulation.

In Lach and Lübbecke (2012), the problem is decomposed into two stages: the first finds a feasible time-schedule, the second assigns the rooms to the lectures. This is based on their work in Lach and Lübbecke (2008) where they considered only hard constraints and showed that specific cuts could be added to the first stage problem to ensure feasible solutions. In Lach and Lübbecke (2012), they show how the **RC** constraint can be included in the first stage problem leaving only the **RStab** constraint for the second stage.

In Hao and Benlic (2011), the focus is on obtaining lower bounds for the first stage problem from Lach and Lübbecke (2012). Their approach is to decompose the courses and compute lower bounds for each of the decomposed sets of courses. A lower bound for the original problem is then calculated as the sum of the lower bounds of the decomposed sets of courses. The selection of the decomposition is embedded in a Tabu Search procedure (see Glover and Laguna, 2013).

Asín Achá and Nieuwenhuis (2012) encodes the constraints as satisfiability clauses. They apply different encodings both treating all constraints as hard and relaxing some or all of the soft constraints. They are able to prove optimality of more than half of the ITC2007 data sets.

Finally, Cacchiani et al (2013) consider a model based on column generation. The idea is to split up the problem into two subproblems. One subproblem focuses on the room related soft constraints **RC** and **RO**. The other subproblem focuses on the time related soft constraints **MWD** and **IL**. The overall lower bound is then the sum of the lower bounds obtained on the subproblems. They are able to obtain improved lower bounds and for three of the instances, they prove that the best known solutions are optimal.

Our approach is similar to Burke et al (2008) in the sense that we enumerate all patterns for each day. The difference is that we apply them for the courses instead of the curricula and include them as variables instead of cuts. This gives us the possibility to make some preprocessing that may not be possible on the compact formulation and to derive valid inequalities from a conflict graph build on the pattern variables.

2 Pattern-based formulation

In this section we present the pattern-based formulation. First the notation used throughout the article is given and then the mathematical model is described.

2.1 Notation

Throughout this article we refer to the following sets: $\mathcal{C}$ (courses), $\mathcal{L}$ (lecturers), $\mathcal{Q}$ (curricula), $\mathcal{D}$ (days), and $\mathcal{T}$ (time slots). Furthermore let $\mathcal{P}$ denote the set of periods, i.e., $\mathcal{P} = \mathcal{D} \times \mathcal{T}$. Then for each day $d \in \mathcal{D}$, the set $\mathcal{P}_d$ denotes the periods that belong to day d. The set of courses in curriculum $q \in \mathcal{Q}$ is $\mathcal{C}_q$, and similarly for a course $c \in \mathcal{C}$, the set of curricula that it belongs to is denoted $\mathcal{Q}_c$. The set of courses taught by lecturer $l \in \mathcal{L}$ is $\mathcal{C}_l$ and $l(c) \in \mathcal{L}$ refers to the lecturer teaching course c.

We also define a set Γ of course cliques. Consider a so-called *course conflict graph* that contains a node for every course. Two nodes are connected by an edge if they cannot be scheduled in the same period, i.e., if they are taught by the same lecturer or are in the same curriculum. As an example, consider the four courses c_1, c_2, c_3 and c_4. Let there be a curriculum consisting of the courses c_1, c_2, and c_3 and let there be a lecturer teaching courses c_3 and c_4. The corresponding course conflict graph is illustrated in Figure 1. All maximal cliques in Γ are enumerated (see Bron and Kerbosch, 1973), i.e., a clique $\gamma \in \Gamma$ is a set of courses $\mathcal{C}_\gamma$ where for each period at most one of the courses can have a lecture scheduled. Let $\mathcal{N}_c \subseteq \mathcal{C} \setminus \{c\}$ be the set of courses that is conflicting with $c \in \mathcal{C}$. In the conflict graph this corresponds to the neighborhood of the node representing c, i.e. all the nodes that are adjacent to c. As an example consider the node representing c_1 in Figure 1. The neighborhood $\mathcal{N}_{c_1}$ then consists of the nodes representing c_2 and c_3. Note that for any course $c_1 \in \mathcal{C}$ and $c_2 \in \mathcal{N}_{c_1}$ it must hold that $c_1 \in \mathcal{N}_{c_2}$.

For a course $c \in \mathcal{C}$, the minimum number of days requested is denoted by the parameter $D_c^{\min}$. If course c is available in a specific period $p = (d, t) \in \mathcal{P}$, then we set the parameter $F_{c,d,t} = F_{c,p} = 1$ otherwise zero,

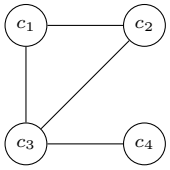

Figure 1: Conflict graph where a curriculum consists of courses c_1, c_2 and c_3 while c_3 and c_4 are taught by the same lecturer.

and $\mathcal{P}_c$ refers to the set of periods where $F_{c,p} = 1$, i.e.,

$$\mathcal{P}_c = \{p \in \mathcal{P} \,|\, F_{c,p} = 1\}\,.$$

The number of lectures that needs to be scheduled for course c is denoted L_c and the total number of lectures to schedule for a clique $\gamma \in \Gamma$ is denoted L_γ, i.e.,

$$L_\gamma = \sum_{c \in \mathcal{C}_\gamma} L_c.$$

Lastly let R denote the number of available rooms and $W^{\mathbf{IL}}$ and $W^{\mathbf{MWD}}$ be the non-negative weights of violating the constraints **IL** and **MWD**, respectively. For $x \in \mathbb{R}$, let the function $(x)^+$ be defined by $(x)^+ := \max\{0, x\}$.

2.2 Mathematical model

The pattern-based formulation aims at assigning a pattern to each day for each course. Given a pattern, the number of lectures scheduled in a pattern is independent on which day it is scheduled to. Therefore we generate all possible patterns based on the time slots $\mathcal{T}$. Consider the case $|\mathcal{T}| = 4$: sixteen different patterns exist which are all illustrated in Table 1. The first column refers to the time slots $t \in \mathcal{T}$ and the remaining columns refer to the patterns $k \in \mathcal{K}$. The symbol "×" in a cell means that the pattern contains the corresponding time slot. The last row (L_k) counts how many lectures pattern k contains.

Table 1: The patterns when $|\mathcal{T}| = 4$.

time slot	pattern index k															
t	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1		×				×	×	×				×	×	×		×
2			×			×			×	×		×	×		×	×
3				×			×		×		×	×		×	×	×
4					×			×		×	×		×	×	×	×
L_k	0	1	1	1	1	2	2	2	2	2	2	3	3	3	3	4

For a pattern $k \in \mathcal{K}$ and a time slot $t \in \mathcal{T}$, the parameter a_t^k is set to one if pattern k contains a lecture in time slot t and zero otherwise. The set $\mathcal{P}_d^k \subseteq \mathcal{P}$ is the set of periods that some course is scheduled in if it is assigned to pattern k at day d. Based on this set, we need to define the feasible patterns $\mathcal{K}_{c,d}$ for each course c and for each day d:

$$\mathcal{K}_{c,d} = \left\{k \in \mathcal{K} \,|\, a_t^k \le F_{c,d,t}\ \forall t \in \mathcal{T}\right\}, \quad c \in \mathcal{C}, d \in \mathcal{D}.$$

For each course $c \in \mathcal{C}$, day $d \in \mathcal{D}$, and pattern $k \in \mathcal{K}_{c,d}$, let $\lambda_{c,d}^k$ be a binary variable taking value one if course c is assigned to pattern k at day d, zero otherwise. For each curriculum $q \in \mathcal{Q}$, day $d \in \mathcal{D}$, and time slot $t \in \mathcal{T}$, let binary variable $s_{q,d,t}$ take value one if curriculum q has an isolated lecture at day d in time slot t, zero otherwise. Finally, for each course $c \in \mathcal{C}$, integer variable w_c counts the number of days below the requested minimum that course c is scheduled.

The pattern-based formulation of the curriculum-based course timetabling problem is as follows.

$$\min W^{\mathbf{IL}} \sum_{q\in\mathcal{Q},d\in\mathcal{D},t\in\mathcal{T}} s_{q,d,t} + W^{\mathbf{MWD}} \sum_{c\in\mathcal{C}} w_c \tag{1}$$

$$\text{s.t.} \sum_{k\in\mathcal{K}_{c,d}} \lambda^k_{c,d} = 1, \quad \forall c \in \mathcal{C}, d \in \mathcal{D} \tag{2}$$

$$\sum_{d\in\mathcal{D},k\in\mathcal{K}_{c,d}} L_k\, \lambda^k_{c,d} = L_c, \quad \forall c \in \mathcal{C} \tag{3}$$

$$\sum_{c\in\mathcal{C}_\gamma,k\in\mathcal{K}_{c,d}} a^k_t\, \lambda^k_{c,d} \le 1, \quad \forall \gamma \in \Gamma, d \in \mathcal{D}, t \in \mathcal{T} \tag{4}$$

$$\sum_{c\in\mathcal{C},k\in\mathcal{K}_{c,d}} a^k_t\, \lambda^k_{c,d} \le R, \quad \forall d \in \mathcal{D}, t \in \mathcal{T} \tag{5}$$

$$\sum_{d\in\mathcal{D},k\in\mathcal{K}_{c,d}:L_k\ge 1} \lambda^k_{c,d} + w_c \ge D^{\min}_c, \quad \forall c \in \mathcal{C} \tag{6}$$

$$\sum_{c\in\mathcal{C}_q,k\in\mathcal{K}_{c,d}} \left(a^k_t - \max\left\{a^k_{t-1}, a^k_{t+1}\right\}\right) \lambda^k_{c,d} \le s_{q,d,t}, \forall q \in \mathcal{Q}, d \in \mathcal{D}, t \in \mathcal{T} \tag{7}$$

$$\lambda^k_{c,d} \in \mathbb{B}, \quad \forall c \in \mathcal{C}, d \in \mathcal{D}, k \in \mathcal{K}_{c,d} \tag{8}$$

$$s_{q,d,t} \in \mathbb{B}, \quad \forall q \in \mathcal{Q}, d \in \mathcal{D}, t \in \mathcal{T} \tag{9}$$

$$w_c \in \mathbb{N}_0, \quad \forall c \in \mathcal{C} \tag{10}$$

The objective function (1) seeks to minimize a weighted sum of the violations of the soft constraints **IL** and **MWD**. Constraints (2) ensure that exactly one pattern is chosen for each course and day, whereas constraints (3) enforce the scheduling of all lectures of all courses. For every clique $\gamma \in \Gamma$, at most one lecture can be scheduled in any period as stipulated by constraints (4). The total number of rooms available is imposed by constraints (5). The number of violations for the soft constraints **IL** and **MWD** are computed through constraints (6) and (7), respectively. Finally, binary and integrality requirements (8)–(10) restrict the domains of the decision variables.

3 Preprocessing

In this section we show how some variables can be removed from the model and specifically how some patterns can never be feasible for some days and courses.

3.1 Simple reductions

Some simple reductions can be made to model (1)–(10). Since a course $c \in \mathcal{C}$ must have L_c lectures scheduled, these require at least $\lceil L_c/|\mathcal{T}|\rceil$ days. So if the minimum working days requested for a course c is less than or equal to this number, i.e., $D^{\min}_c \le \lceil L_c/|\mathcal{T}|\rceil$, the constraint **MWD** can never be violated by that course and we can remove the variable w_c and the associated constraint (6). Another reduction is to consider a course c where $L_c = D^{\min}_c$, i.e., where the requested number of minimum working days is equal to the number of lectures. The only way for the course to not violate the constraint **MWD** is to only use the patterns containing at most one lecture. Assume that the course has chosen one lecture for each day. If the course then chooses to increase the number of lectures for one of the days, then it has to remove a lecture from one of the other days thus increasing the violation of the constraint **MWD** by one. This means that we can substitute the variable w_c throughout the model by the sum of the patterns containing at least two lectures

where the coefficients are the number of lectures in the patterns minus one:

$$w_c = \sum_{\substack{d\in\mathcal{D},\\ k\in\mathcal{K}_{c,d}:\\ L_k\geq 2}} (L_k - 1)\ \lambda^k_{c,d}, \quad \forall c \in \mathcal{C} : L_c = D^{\min}_c. \tag{11}$$

Another special case is when course c requests two working days, i.e., that $D^{\min}_c = 2$. The only way that the constraint **MWD** can be violated is when all lectures of the course are scheduled in a single day. Therefore the variable w_c can be substituted throughout the model by the sum of the patterns scheduling all lectures on one day:

$$w_c = \sum_{d\in\mathcal{D}, k\in\mathcal{K}_{c,d}: L_k = L_c} \lambda^k_{c,d}, \quad \forall c \in \mathcal{C} : D^{\min}_c = 2. \tag{12}$$

Some of the variables used for calculating the isolated lectures can also be substituted. Consider some curriculum $q \in \mathcal{Q}$, day $d \in \mathcal{D}$ and time slot $t \in \mathcal{T}$, and let $\mathcal{C}_{q,d,t}$ be the set of courses in the curriculum that can be scheduled in either that period or an adjacent period:

$$\mathcal{C}_{q,d,t} = \{c \in \mathcal{C}_q \,|\, F_{c,d,t-1} = 1 \vee F_{c,d,t} = 1 \vee F_{c,d,t+1} = 1\}\,. \tag{13}$$

If $|\mathcal{C}_{q,d,t}| = 1$ and the associated course chooses a pattern where a lecture is scheduled at day d in time slot t but not in an adjacent period, then this lecture must be an isolated lecture for q. This means we can substitute the variable $s_{q,d,t}$ in the objective function (1):

$$s_{q,d,t} = \sum_{\substack{c\in\mathcal{C}_q, k\in\mathcal{K}_{c,d}:\\ a^k_t=1\wedge a^k_{t-1}=a^k_{t+1}=0}} \lambda^k_{c,d}, \quad \forall q \in \mathcal{Q}, d \in \mathcal{D}, t \in \mathcal{T} : |\mathcal{C}_{q,d,t}| = 1. \tag{14}$$

3.2 Pattern elimination

When all patterns are generated, not all of them need to be considered for some course $c \in \mathcal{C}$. The total number of lectures to schedule for c is L_c. Clearly if some pattern contains more than L_c lectures, then this pattern can be excluded from all days. Furthermore, suppose that a pattern $k \in \mathcal{K}_{c,d}$ for some day $d \in \mathcal{D}$ has been selected. This means that no more patterns can be selected for day d so the remaining patterns for the days $\mathcal{D}\backslash\{d\}$ must be able to cover the remaining lectures $L_c - L_k$:

$$\sum_{d'\in\mathcal{D}\backslash\{d\}} \max_{k'\in\mathcal{K}_{c,d'}} \{L_{k'}\} \geq L_c - L_k, \quad \forall d \in \mathcal{D}, c \in \mathcal{C}, k \in \mathcal{K}_{c,d}. \tag{15}$$

If for any day $d \in \mathcal{D}$ and pattern $k \in \mathcal{K}_{c,d}$ the condition (15) is not met, then this pattern is removed from that day. Similarly if for all days except d, patterns are chosen to cover a minimum number of lectures, then any pattern in $\mathcal{K}_{c,d}$ cannot cover more than the remaining lectures:

$$\sum_{d'\in\mathcal{D}\backslash\{d\}} \min_{k'\in\mathcal{K}_{c,d'}} \{L_{k'}\} \leq L_c - L_k, \quad \forall d \in \mathcal{D}, c \in \mathcal{C}, k \in \mathcal{K}_{c,d}. \tag{16}$$

So we scan through all patterns and remove the ones that do not fulfill conditions (15) and (16). If any patterns are removed during this scan, we iterate until no more patterns are removed.

The next elimination considers a course and checks if choosing a pattern would overlap too many feasible periods for some other courses. Consider the course $c_1 \in \mathcal{C}$ and the conflicting course $c_2 \in \mathcal{N}_{c_1}$. Course c_1 can at most be scheduled in $|\mathcal{P}_{c_2}| - L_{c_2}$ of the periods that are feasible for c_2 since all lectures must be scheduled:

$$\left|\mathcal{P}_{c_2} \cap \mathcal{P}^k_d\right| \leq |\mathcal{P}_{c_2}| - L_{c_2}, \quad \forall d \in \mathcal{D}, c_1 \in \mathcal{C}, c_2 \in \mathcal{N}_{c_1}, k \in \mathcal{K}_{c_1,d}. \tag{17}$$

The left-hand side in (17) is the number of periods that are feasible for c_2 and occupied by c_1 if pattern k is chosen. The right-hand side is the total number of feasible periods for c_2 minus the number of lectures

that must be scheduled. Any pattern not fulfilling condition (17) is then removed from the model. This can be extended to the course cliques. Consider clique $\gamma \in \Gamma$ and a course $c \in \mathcal{C}_\gamma$ and let $\mathcal{P}_{\gamma\backslash c}$ be the feasible periods for all the courses in $\mathcal{C}_\gamma$ except c: $\mathcal{P}_{\gamma\backslash c} = \bigcup_{c' \in \mathcal{C}_\gamma \backslash \{c\}} \mathcal{P}_{c'}$. Any pattern chosen by c may not occupy more than the number of periods in $\mathcal{P}_{\gamma\backslash c}$ minus the sum of all the lectures to schedule for the courses in the clique except c since all lectures must be scheduled:

$$\left|\mathcal{P}_{\gamma\backslash c} \cap \mathcal{P}_d^k\right| \leq \left|\mathcal{P}_{\gamma\backslash c}\right| - \sum_{c' \in \mathcal{C}_\gamma \backslash \{c\}} L_{c'}, \quad \forall d \in \mathcal{D}, \gamma \in \Gamma, c \in \mathcal{C}_\gamma, k \in \mathcal{K}_{c,d}. \tag{18}$$

The left-hand side in (18) is the number of periods in the union of all feasible periods for the courses in the clique except c which is occupied by c if pattern k is selected. The right-hand side is the size of the union of the feasible periods of all the courses in the clique except c minus the total number of lectures that must be scheduled for these courses. Any pattern not fulfilling (18) are then removed.

The last elimination procedure solves a series of maximum flow problems. The idea is to consider a specific pattern for some course and then see if the remaining lectures in some clique can be scheduled. Consider a clique $\gamma \in \Gamma$, a pattern $k \in \mathcal{K}_{c,d}$ for some course $c \in \mathcal{C}_\gamma$ and day $d \in \mathcal{D}$. Let $m = L_k$, $n = |\mathcal{P}|$, and assume without loss of generality that the periods are numbered such that the first m periods, p_1, p_2, …, p_m, are those contained in pattern k. Let the next $n - m$ periods p_{m+1}, p_{m+2}, …, p_n be the remaining ones, i.e., all the periods that are not contained in the pattern. Since the course must choose exactly one pattern for each day, we know that if pattern k is selected then the course cannot be scheduled in the remaining periods on day d. Furthermore since we are considering a clique, we know that only the course under consideration can be scheduled in the first m periods. Assume that the courses are numbered such that the first $|\mathcal{C}_\gamma|$ are the courses that are in clique γ: c_1, c_2, …, $c_{|\mathcal{C}_\gamma|}$. Create a graph with a source node $\mathfrak{s}$ and a sink node $\mathfrak{t}$ (see Figure 2). An additional node $\mathfrak{s}$' is created to act as a dummy source of the non-selected periods. An arc with capacity $L_\gamma - L_k$ is added from the source $\mathfrak{s}$ to the dummy source $\mathfrak{s}$' to model that the first L_k lectures must be scheduled by the source and the remaining $L_\gamma - L_k$ lectures must be scheduled by the dummy source. For each period p_1, p_2, …, p_n, create a node and add an ingoing arc from the source $\mathfrak{s}$ if the period is in the set $\{p_1, p_2, \ldots, p_m\}$ and add an ingoing arc from the dummy source $\mathfrak{s}$' if the period is in the set $\{p_{m+1}, p_{m+2}, \ldots, p_n\}$. The capacities of all these arcs are set to one. For each course $c' \in \{c_1, c_2, \ldots, c_{|\mathcal{Q}_\gamma|}\}$, add a node to the graph and add an outgoing arc to the sink $\mathfrak{t}$ with a capacity of $L_{c'}$. For each node $p \in \{p_1, p_2, \ldots, p_n\}$ and each node $c' \in \{c_1, c_2, \ldots, c_{|\mathcal{C}_\gamma|}\}$, add an arc from p to c' with capacity $a(c', p)$ which is set depending on c and k:

$$a(c', p) = \begin{cases} F_{c',p} & \text{if } p \in \{p_1, \ldots, p_m\} \\ & \quad \text{and } c' = c \\ F_{c',p} & \text{if } p \in \{p_{m+1}, \ldots, p_n\} \\ & \quad \text{and } (c' \neq c \vee p \notin P_d) \\ 0 & \text{otherwise.} \end{cases} \tag{19}$$

If p is one of the first m periods, the capacity is set to $F_{c',p}$ if $c' = c$; if p is one of the $n - m$ last periods, the capacity is set to $F_{c',p}$ if p does not belong to day d or if $c' \neq c$. A maximum flow problem is solved and if the total amount of flow is less than the total number of lectures that need to be scheduled for the clique, then pattern k can never be feasible for course c at day d, and can thus be removed from $\mathcal{K}_{c,d}$.

4 Valid inequalities

In this section some strengthened bounds and valid inequalities are presented.

4.1 Implied bounds

Burke et al (2012) noted that since all lectures must be scheduled, then at least one day must be used by each course $c \in \mathcal{C}$ meaning that the constraint **MWD** can at most be violated by $D_c^{\min} - 1$. This can be

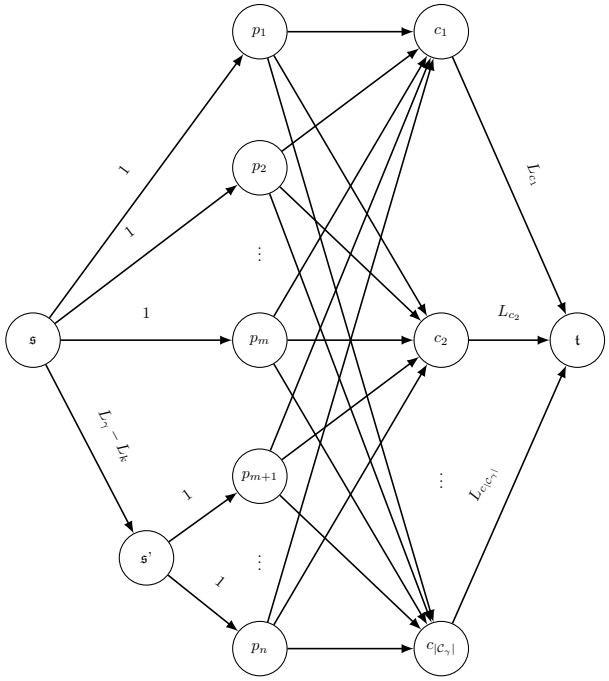

Figure 2: Given a clique γ, maximum flow graph to verify if pattern k for course $c \in \mathcal{C}_\gamma$ and day d can be discarded.

strengthened a bit by noting that at most $|\mathcal{T}|$ lectures can be scheduled every day and so course c must be scheduled in at least $\lceil L_c/|\mathcal{T}| \rceil$ days:

$$0 \leq w_c \leq \left(D_c^{\min} - \left\lceil \frac{L_c}{|T|} \right\rceil \right)^+, \quad \forall c \in \mathcal{C}. \tag{20}$$

By considering the constraint **A**, this can be slightly improved. Let $\mathcal{D}_c = \left\{ d_1, d_2, \ldots, d_{|\mathcal{D}|} \right\}$ be the days for course c which are sorted such that $\sum_{t \in \mathcal{T}} F_{c,d_i,t} \geq \sum_{t \in \mathcal{T}} F_{c,d_j,t}$ for every $i, j \in \{1, 2, \ldots, |\mathcal{D}|\}$ where $i < j$. Then the minimum number of days that course c will be scheduled is the smallest index $i \in \{1, 2, \ldots, |\mathcal{D}|\}$ such that the total amount of feasible periods in the days $d_1, d_2, \ldots, d_i$ is enough to schedule all lectures of the course. We can then calculate an upper bound on the violation of the constraint **MWD**:

$$w_c^u = D_c^{\min} - \min_{i \in \{1,\ldots,|\mathcal{D}|\}} \left\{ i \left| \sum_{\substack{j \in \{1,\ldots,i\}, \\ p \in \mathcal{P}_{d_j}}} F_{c,p} \geq L_c \right. \right\}, \quad \forall c \in \mathcal{C}. \tag{21}$$

It is also possible to calculate the maximum number of days course c can be scheduled in. Obviously it is not possible to schedule more than L_c days. Using the constraint **A**, it is not possible to schedule more days than the set of days where at least one of the time slots is feasible for the course. So a lower bound on the violation of the constraint **MWD** can also be calculated:

$$w_c^l = D_c^{\min} - \max \left\{ L_c, \left| \left\{ d \in \mathcal{D} \,\middle|\, \sum_{p \in \mathcal{P}_d} F_{c,p} \geq 1 \right\} \right| \right\}, \quad \forall c \in \mathcal{C}. \tag{22}$$

This gives the following bounds:

$$\left(w_c^l\right)^+ \leq w_c \leq \left(w_c^u\right)^+, \quad \forall c \in \mathcal{C}. \tag{23}$$

Furthermore since we have just found the minimum and maximum number of days that a course can have scheduled lectures, we can add the following inequalities to model (1)–(10):

$$\sum_{\substack{d\in\mathcal{D},\\ k\in\mathcal{K}_{c,d}:\\ L_k\geq 1}} \lambda^k_{c,d} \geq \min_{i\in\{1,\ldots,|\mathcal{D}|\}} \left\{ i \;\middle|\; \sum_{\substack{j\in\{1,\ldots,i\},\\ p\in\mathcal{P}_{d_j}}} F_{c,p} \geq L_c \right\}, \quad \forall c\in\mathcal{C}. \tag{24}$$

$$\sum_{\substack{d\in\mathcal{D},\\ k\in\mathcal{K}_{c,d}:\\ L_k\geq 1}} \lambda^k_{c,d} \leq \max \left\{ L_c, \left| \left\{ d\in\mathcal{D} \,|\, \sum_{p\in\mathcal{P}_d} F_{c,p} \geq 1 \right\} \right| \right\}, \quad \forall c\in\mathcal{C}. \tag{25}$$

4.2 Extended cover inequalities

Consider clique $\gamma\in\Gamma$. Since we must schedule the lectures for all courses, trivially the following must hold:

$$\sum_{c\in\mathcal{C}_\gamma, d\in\mathcal{D}, k\in\mathcal{K}_{c,d}} L_k\,\lambda^k_{c,d} \leq L_\gamma, \quad \forall\gamma\in\Gamma. \tag{26}$$

Note that the conditions (26) are knapsack constraints. This means that we can apply known valid inequalities such as *cover inequalities*, see e.g. Van Roy and Wolsey (1987). As an example, consider some clique $\gamma\in\Gamma$ with $L_\gamma = 5$. Assume we only choose patterns containing two lectures, then surely we cannot choose more than two of these patterns otherwise the knapsack constraint is violated. This means that if we choose three patterns, e.g. $\lambda^{k_1}_{c_1,d_1}$, $\lambda^{k_2}_{c_2,d_2}$ and $\lambda^{k_3}_{c_3,d_3}$ where $L_{k_1} = L_{k_2} = L_{k_3} = 2$, then we have a cover and we can add the *cover inequality*:

$$\lambda^{k_1}_{c_1,d_1} + \lambda^{k_2}_{c_2,d_2} + \lambda^{k_3}_{c_3,d_3} \leq 2.$$

We can then add each variable to the inequality where $L_k \geq 2$ creating an *extended cover inequality*. This can be generalized to the following inequalities:

$$\sum_{\substack{c\in\mathcal{C}_\gamma, d\in\mathcal{D},\\ k\in\mathcal{K}_{c,d}: L_k\geq i}} \lambda^k_{c,d} \leq \left\lfloor \frac{L_\gamma}{i} \right\rfloor, \quad \forall\gamma\in\Gamma, i\in\left\{2,\ldots,\left\lfloor\frac{L_\gamma}{2}\right\rfloor+1\right\}. \tag{27}$$

These inequalities consider that each clique $\gamma\in\Gamma$ can at most be scheduled for L_γ lectures. Since all lectures must be scheduled, L_γ is also a lower bound on the number of lectures that must be scheduled:

$$\sum_{c\in\mathcal{C}_\gamma, d\in\mathcal{D}, k\in\mathcal{K}_{c,d}} L_k\,\lambda^k_{c,d} \geq L_\gamma, \quad \forall\gamma\in\Gamma. \tag{28}$$

To be able to add the *extended cover inequalities* by using constraints (28), we need to reformulate it into a knapsack constraint similar to (26). To do this, we first define $L^{\max}_{c,d}$ as the maximum number of lectures that course $c\in\mathcal{C}$ can be scheduled on day $d\in\mathcal{D}$:

$$L^{\max}_{c,d} := \max_{k\in\mathcal{K}_{c,d}} \{L_k\}. \tag{29}$$

Taking constraints (2) and multiplying by $L^{\max}_{c,d}$ gives us the following:

$$\sum_{k\in\mathcal{K}_{c,d}} L^{\max}_{c,d}\,\lambda^k_{c,d} = L^{\max}_{c,d}, \quad \forall c\in\mathcal{C}, d\in\mathcal{D}. \tag{30}$$

Obviously we can add $L_k - L_k$ to the coefficient of $\lambda^k_{c,d}$ in (30) and rearrange the terms:

$$\sum_{k\in\mathcal{K}_{c,d}} \left(L_k + L^{\max}_{c,d} - L_k\right)\,\lambda^k_{c,d} = L^{\max}_{c,d}, \quad \forall c\in\mathcal{C}, d\in\mathcal{D}. \tag{31}$$

$$\sum_{k \in \mathcal{K}_{c,d}} L_k \, \lambda_{c,d}^k = L_{c,d}^{\max} - \sum_{k \in \mathcal{K}_{c,d}} \left(L_{c,d}^{\max} - L_k \right) \lambda_{c,d}^k, \quad \forall c \in \mathcal{C}, d \in \mathcal{D}. \tag{32}$$

Substituting into (28) and rearranging the terms gives the following:

$$\sum_{c \in \mathcal{C}_\gamma, d \in \mathcal{D}} \left(L_{c,d}^{\max} - \sum_{k \in \mathcal{K}_{c,d}} \left(L_{c,d}^{\max} - L_k \right) \lambda_{c,d}^k \right) \geq L_\gamma, \quad \forall \gamma \in \Gamma. \tag{33}$$

$$\sum_{c \in \mathcal{C}_\gamma, d \in \mathcal{D}, k \in \mathcal{K}_{c,d}} \left(L_{c,d}^{\max} - L_k \right) \lambda_{c,d}^k \leq \sum_{c \in \mathcal{C}_\gamma, d \in \mathcal{D}} L_{c,d}^{\max} - L_\gamma, \quad \forall \gamma \in \Gamma. \tag{34}$$

Since $L_{c,d}^{\max}$ is the maximum number of lectures that can be scheduled for course c on day d, then $L_{c,d}^{\max} \geq L_k$ for every pattern $k \in \mathcal{K}_{c,d}$. Furthermore we assume that $\sum_{d \in \mathcal{D}} L_{c,d}^{\max} \geq L_c$ for each course c, otherwise it can easily be seen that the problem would be infeasible. Hence constraints (34) are the wanted knapsack constraints. For ease of notation, we define:

$$\overline{L}_c^k := L_{c,d}^{\max} - L_k, \quad \forall c \in \mathcal{C}, d \in \mathcal{D}, k \in \mathcal{K}_{c,d}. \tag{35}$$

$$\overline{L}_\gamma := \sum_{c \in \mathcal{C}_\gamma, d \in \mathcal{D}} L_{c,d}^{\max} - L_\gamma, \quad \forall \gamma \in \Gamma. \tag{36}$$

We can then add the *extended cover inequalities* in the same way as the inequalities (27) for constraints (26):

$$\sum_{\substack{c \in \mathcal{C}_\gamma, d \in \mathcal{D}, \\ k \in \mathcal{K}_{c,d} : \overline{L}_c^k \geq i}} \lambda_{c,d}^k \leq \left\lfloor \frac{\overline{L}_\gamma}{i} \right\rfloor, \quad \forall \gamma \in \Gamma, i \in \left\{ 2, \ldots, \left\lfloor \frac{\overline{L}_\gamma}{2} \right\rfloor + 1 \right\}. \tag{37}$$

Not all of *extended cover inequalities* are necessarily added. Consider some clique $\gamma \in \Gamma$ and $i, j \in \{2, \ldots, \lfloor L_\gamma/2 \rfloor + 1\}$ where $i < j$. If $\lfloor L_\gamma/i \rfloor = \lfloor L_\gamma/j \rfloor$, constraint (27) with respect to j is redundant and is not added to the model. The same argument can be used for constraints (37).

4.3 The pattern conflict graph and inequalities

The inequalities we focus on in this section are based on creating a *pattern conflict graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where for each course $c \in \mathcal{C}$, day $d \in \mathcal{D}$ and pattern $k \in \mathcal{K}_{c,d}$, we create a node $v_{c,d}^k \in \mathcal{V}$. Two nodes $v_{c_1,d_1}^{k_1}, v_{c_2,d_2}^{k_2} \in \mathcal{V}$ are connected with an edge if the corresponding patterns cannot both be included in a feasible solution, i.e., if they are *conflicting*. We then use this graph to derive valid inequalities. Some of these inequalities are clique inequalities derived by finding the cliques in the pattern conflict graph. This graph can become very large so it is not practical to enumerate all the maximal cliques as we did in Section 2 for the *course conflict graph*. Instead we run a heuristic aimed at minimizing the number of cliques needed to cover all edges as described by Kou et al (1978). Let Θ be the set of cliques returned by the heuristic and let $\mathcal{V}_\theta \subseteq \mathcal{V}$ be the set of nodes in clique $\theta \in \Theta$. We then add the following clique inequalities:

$$\sum_{v_{c,d}^k \in \mathcal{V}_\theta} \lambda_{c,d}^k \leq 1, \quad \forall \theta \in \Theta. \tag{38}$$

The next set of valid inequalities is based on constraints (7) and inspired by the *Lifted Class Compactness* inequalities described in Avella and Vasil'Ev (2005, Proposition 5.4). In the left-hand side of constraint (7) for curriculum $q \in \mathcal{Q}$ and time slot $t \in \mathcal{T}$, let $\overline{a}_t^k$ be the coefficient of the pattern variable $\lambda_{c,d}^k$ for course $c \in \mathcal{C}_q$, day $d \in \mathcal{D}$ and pattern $k \in \mathcal{K}_{c,d}$:

$$\begin{aligned}\overline{a}_t^k :=& a_t^k - \max\{a_{t-1}^k, a_{t+1}^k\} \\ =& \begin{cases} 1 & \text{if } a_t^k = 1 \wedge a_{t-1}^k = a_{t+1}^k = 0 \\ -1 & \text{if } a_t^k = 0 \wedge \left(a_{t-1}^k = 1 \vee a_{t+1}^k = 1\right) \\ 0 & \text{otherwise.} \end{cases}\end{aligned} \tag{39}$$

Let $\mathcal{V}^+_{q,d,t}$ and $\mathcal{V}^-_{q,d,t}$ be the sets of nodes representing the patterns for which the coefficient is equal to one and to minus one, respectively:

$$\mathcal{V}^+_{q,d,t} := \left\{v_{c,d}^k \in \mathcal{V} \,|\, c \in \mathcal{C}_q, k \in \mathcal{K}_{c,d}, \overline{a}_t^k = 1\right\}. \tag{40}$$

$$\mathcal{V}^-_{q,d,t} := \left\{v_{c,d}^k \in \mathcal{V} \,|\, c \in \mathcal{C}_q, k \in \mathcal{K}_{c,d}, \overline{a}_t^k = -1\right\}. \tag{41}$$

Based on these sets, we can reformulate constraints (7):

$$\sum_{v_{c,d}^k \in \mathcal{V}^+_{q,d,t}} \lambda_{c,d}^k - \sum_{v_{c,d}^k \in \mathcal{V}^-_{q,d,t}} \lambda_{c,d}^k \leq s_{q,d,t}, \quad \forall q \in \mathcal{Q}, d \in \mathcal{D}, t \in \mathcal{T}. \tag{42}$$

In constraints (42), it can be seen that for a curriculum $q \in \mathcal{Q}$, there is an isolated lecture at day d and time slot t if one of the patterns in $\mathcal{V}^+_{q,d,t}$ is selected and none of the patterns in $\mathcal{V}^-_{q,d,t}$ are selected. Consider now some clique $\mathcal{H}_{q,d,t} \subseteq \mathcal{V}$, where every node in $\mathcal{H}_{q,d,t}$ is conflicting with every node in $\mathcal{V}^-_{q,d,t}$. If any pattern in $\mathcal{H}_{q,d,t}$ is included in a solution, then none of the patterns in $\mathcal{V}^-_{q,d,t}$ can be chosen. This means that if a pattern has been chosen from the set $\mathcal{V}^+_{q,d,t}$ and a pattern has been chosen from the set $\mathcal{H}_{q,d,t}$, then curriculum q must have an isolated lecture at day d and time slot t. Therefore the following *adjacent isolated lecture inequalities* can be added:

$$\sum_{v_{c,d}^k \in \mathcal{V}^+_{q,d,t}} \lambda_{c,d}^k + \sum_{v_{c,d'}^k \in \mathcal{H}_{q,d,t}} \lambda_{c,d}^k - 1 \leq s_{q,d,t}, \quad \forall q \in \mathcal{Q}, d \in \mathcal{D}, t \in \mathcal{T}. \tag{43}$$

The way we generate a clique $\mathcal{H}_{q,d,t}$ is to take the subgraph induced by the neighbors of $\mathcal{V}^-_{q,d,t}$. We then run the heuristic described by Kou et al (1978) to get a clique cover of the neighbors and, for each clique returned by the heuristic, an inequality (43) is added, where the clique defines set $\mathcal{H}_{q,d,t}$.

4.4 Generating edges for the pattern conflict graph

The pattern conflict graph was briefly described in Section 4.3. Here we explain how we derive the edges, i.e., how we figure out which of the patterns are conflicting. The first set of edges are derived from constraints (2). Since at most one pattern can be chosen for each course and each day, we can add an edge between nodes $v_{c,d}^{k_1}$ and $v_{c,d}^{k_2}$ for each course $c \in \mathcal{C}$, day $d \in \mathcal{D}$, and patterns $k_1 \in \mathcal{K}_{c,d}$ and $k_2 \in \mathcal{K}_{c,d} \backslash \{k_1\}$. Next we can also add edges between nodes $v_{c,d_1}^{k_1}$ and $v_{c,d_1}^{k_2}$ if $L_{k_1} + L_{k_2} > L_c$.

Another set of edges comes from the constraint **C**. For two conflicting courses $c_1, c_2 \in \mathcal{C}$, we add an edge between nodes $v_{c_1,d}^{k_1}$ and $v_{c_2,d}^{k_2}$ for day $d \in \mathcal{D}$ and patterns $k_1 \in \mathcal{K}_{c_1,d}$ and $k_2 \in \mathcal{K}_{c_2,d}$ if there exists some time slot contained in both patterns, i.e., if $\exists t \in \mathcal{T} : a_t^{k_1} = a_t^{k_2} = 1$.

The next set of edges is derived by extending the methods to eliminate the patterns as discussed in Section 3, where instead of checking for infeasibility of selecting only one pattern, we check for infeasibility of selecting a pair of patterns. For example, it was checked in constraint (15) whether selecting a specific pattern would make it possible to cover the remaining lectures on the other days. Consider some course $c \in \mathcal{C}$, two days $d_1, d_2 \in \mathcal{D}$, and two patterns $k_1 \in \mathcal{K}_{c,d_1}$ and $k_2 \in \mathcal{K}_{c,d_2}$ where $d_1 \neq d_2$. If both patterns are selected, then it must be possible to cover the remaining $L_c - L_{k_1} - L_{k_2}$ lectures from the patterns associated

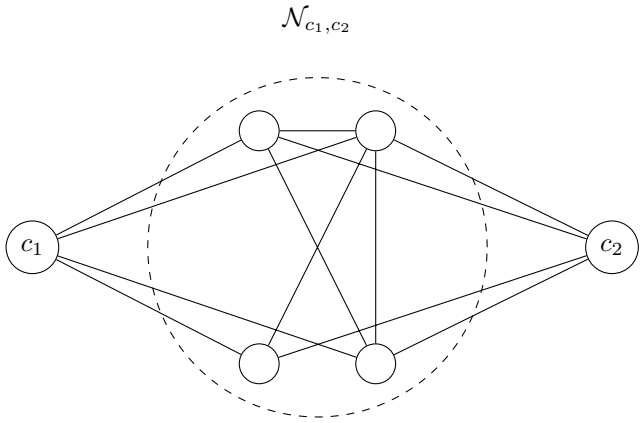

Figure 3: Neighborhood $\mathcal{N}_{c_1,c_2}$ of the courses $c_1, c_2 \in \mathcal{C}$ in the course clique graph.

with the days in set $\mathcal{D}\backslash\{d_1, d_1\}$. We add an edge between nodes $v^{k_1}_{c,d_1}$ and $v^{k_2}_{c,d_2}$ if the following condition is not met:

$$\sum_{d\in\mathcal{D}\backslash\{d_1,d_2\}} \max_{k\in\mathcal{K}_{c,d}} \{L_k\} \geq L_c - L_{k_1} - L_{k_2}. \tag{44}$$

Similarly if the two patterns $k_1 \in \mathcal{K}_{c,d_1}$ and $k_2 \in \mathcal{K}_{c,d_2}$ are chosen, then the patterns containing the smallest number of lectures on the other days cannot exceed the remaining lectures. We add an edge between nodes $v^{k_1}_{c,d_1}$ and $v^{k_2}_{c,d_2}$ if the following condition is not met:

$$\sum_{d\in\mathcal{D}\backslash\{d_1,d_2\}} \min_{k\in\mathcal{K}_{c,d}} \{L_k\} \leq L_c - L_{k_1} - L_{k_2}. \tag{45}$$

We can also verify if selecting a pair of patterns would overlap too many feasible periods for some courses as in (17) and (18). First all pairs of conflicting courses are considered. Let $c_1, c_2 \in \mathcal{C}$ be conflicting. If pattern $k_2 \in \mathcal{K}_{c_2,d_2}$ is chosen for course c_2 on day $d_2 \in \mathcal{D}$, we know that no other pattern can be chosen for course c_2 for that day. Therefore it is not possible to choose a pattern $k_1 \in \mathcal{K}_{c_1,d_1}$ for course c_1 on day $d_1 \in \mathcal{D}\backslash\{d_2\}$ if that pattern occupies more than $L_{c_2} - L_{k_2}$ of the feasible periods for c_2 in the days $\mathcal{D}\backslash\{d_2\}$. Hence, for the pair of nodes $v^{k_1}_{c_1,d_1}$ and $v^{k_2}_{c_2,d_2}$, we add an edge if the following condition does not hold:

$$\left|\mathcal{P}_{c_2} \cap \mathcal{P}^{k_1}_{d_1}\right| \leq |\mathcal{P}_{c_2} \setminus \mathcal{P}_{d_2}| - L_{c_2} + L_{k_2}. \tag{46}$$

Consider now two courses $c_1, c_2 \in \mathcal{C}$ that need not be conflicting nor distinct. In the course conflict graph, we look at the nodes representing c_1 and c_2 and the neighborhood of these two nodes, i.e., the nodes that are connected by an edge to both c_1 and c_2. Let $\mathcal{N}_{c_1,c_2}$ be this neighborhood (illustrated in Figure 3) and defined as

$$\mathcal{N}_{c_1,c_2} = \mathcal{N}_{c_1} \cap \mathcal{N}_{c_2}. \tag{47}$$

It is not possible to assign a pair of patterns to courses c_1 and c_2 that together occupy more than $|\mathcal{P}_{c_3}| - L_{c_3}$ of the feasible periods of $c_3 \in \mathcal{N}_{c_1,c_2}$. Therefore we get the following conditions:

$$\left|\mathcal{P}_{c_3} \cap \left(\mathcal{P}^{k_1}_{d_1} \cup \mathcal{P}^{k_2}_{d_2}\right)\right| \leq |\mathcal{P}_{c_3}| - L_{c_3}. \tag{48}$$

Hence for each pair of nodes $v^{k_1}_{c_1,d_1}$ and $v^{k_2}_{c_2,d_2}$, we add an edge if conditions (48) are not met. This can be extended by considering a pair of nodes in the neighborhood of c_1 and c_2 that are themselves conflicting. Let $c_3, c_4 \in \mathcal{N}_{c_1,c_2}$ be such a pair, i.e., $c_4 \in \mathcal{N}_{c_3}$. It is not feasible to assign to courses c_1 and c_2 patterns that

together occupy more than $|\mathcal{P}_{c_3} \cup \mathcal{P}_{c_4}| - L_{c_3} - L_{c_4}$ of the feasible periods for c_3 and c_4 because all of their lectures must be scheduled in distinct periods. The following conditions must be met:

$$\left|(\mathcal{P}_{c_3} \cup \mathcal{P}_{c_4}) \cap \left(\mathcal{P}_{d_1}^{k_1} \cup \mathcal{P}_{d_2}^{k_2}\right)\right| \leq |\mathcal{P}_{c_3} \cup \mathcal{P}_{c_4}| - L_{c_3} - L_{c_4}. \tag{49}$$

We can extend it even further by considering cliques in the neighborhood of c_1 and c_2. Let Γ_{c_1,c_2} be a set of cliques of the neighborhood $\mathcal{N}_{c_1,c_2}$ found by iterating through each clique $\gamma \in \Gamma$ and then add the courses $\mathcal{C}' = \mathcal{N}_{c_1,c_2} \cap \mathcal{C}_\gamma$ if $\mathcal{C}' \neq \emptyset$:

$$\Gamma_{c_1,c_2} = \{\mathcal{C}' \subseteq \mathcal{N}_{c_1,c_2} \,|\, \exists \gamma \in \Gamma : \mathcal{C}' \subseteq \mathcal{C}_\gamma\}. \tag{50}$$

As for a clique in Γ, we also refer to the courses in $\gamma \in \Gamma_{c_1,c_2}$ by $\mathcal{C}_\gamma$. The set of feasible periods of all the courses in $\mathcal{C}_\gamma$ is denoted by $\mathcal{P}_\gamma^{c_1,c_2}$:

$$\mathcal{P}_\gamma^{c_1,c_2} = \bigcup_{c \in \mathcal{C}_\gamma} \mathcal{P}_c. \tag{51}$$

Note that $\mathcal{P}_\gamma^{c_1,c_2}$ does not necessarily include the feasible periods for c_1 and c_2 as they are not in the set of courses $\mathcal{C}_\gamma$. We add an edge between nodes $v_{c_1,d_2}^{k_1}$ and $v_{c_2,d_2}^{k_2}$ if the following conditions are not met:

$$\left|\mathcal{P}_\gamma^{c_1,c_2} \cap \left(\mathcal{P}_{d_1}^{k_1} \cup \mathcal{P}_{d_2}^{k_2}\right)\right| \leq \left|\mathcal{P}_\gamma^{c_1,c_2}\right| - \sum_{c' \in \mathcal{C}_\gamma} L_{c'}. \tag{52}$$

The last method to identify edges in the pattern conflict graph is to solve a series of maximum flow problems. The maximum flow graph is an extension to the graph used for eliminating patterns in Section 3.2. Consider a clique $\gamma \in \Gamma$, a pattern $k_1 \in \mathcal{K}_{c_1,d_1}$ for some course $c_1 \in \mathcal{C}_\gamma$ and day $d_1 \in \mathcal{D}$ and a pattern $k_2 \in \mathcal{K}_{c_2,d_2}$ for some course $c_2 \in \mathcal{C}_\gamma$ and day $d_2 \in \mathcal{D}$. Let $m_1 = L_{k_1}$, $m_2 = L_{k_2} + L_{k_1}$ and $n = |\mathcal{P}|$ and assume that the periods are numbered such that the first m_1 periods p_1, p_2, …, p_{m_1} are the periods contained in pattern k_1 and the next $m_2 - m_1$ periods p_{m_1+1}, p_{m_1+2}, …, p_{m_2} are the periods contained in pattern k_2. Let the next $n - m_2$ periods p_{m_2+1}, p_{m_2+2}, …, p_n be the remaining periods, i.e., all the periods that are not contained in the patterns. Since exactly one pattern must be chosen for each course c_1 and c_2 and each day, we know that if patterns k_1 and k_2 are selected, then the courses cannot be scheduled in the remaining periods on day d_1 and d_2, respectively. Furthermore since we are considering a clique, we know that only c_1 can be scheduled in the first m_1 periods and only c_2 can be scheduled in the next $m_2 - m_1$ periods. Assume that the courses are numbered such that the first $|\mathcal{C}_\gamma|$ are the courses in clique γ: c_1, c_2, …, $c_{|\mathcal{C}_\gamma|}$. Create a graph with a source node $\mathfrak{s}$, a dummy source node $\mathfrak{s}$', and a sink node $\mathfrak{t}$ (see Figure 4). An arc with capacity $L_\gamma - L_{k_1} - L_{k_2}$ is added from the source $\mathfrak{s}$ to the dummy source $\mathfrak{s}$'. For each period p_1, p_2, …, p_n, create a node and add an ingoing arc from the source $\mathfrak{s}$ if the period is in the set $\{p_1, p_2, \ldots, p_{m_2}\}$ and an ingoing arc from the dummy source $\mathfrak{s}$' if the period is in the set $\{p_{m_2+1}, p_{m_2+2}, \ldots, p_n\}$. The capacities of all these arcs are set to one. For each course $c' \in \{c_1, c_2, \ldots, c_{|\mathcal{Q}_\gamma|}\}$, add a node to the graph and an outgoing arc to the sink $\mathfrak{t}$ with a capacity of $L_{c'}$. For each node $c' \in \{c_1, c_2, \ldots, c_{|\mathcal{C}_\gamma|}\}$ and for each node $p \in \{p_1, p_2, \ldots, p_n\}$, add an arc from p to c' with capacity $a\,(p, c')$ defined as follows:

$$a\,(p, c') = \begin{cases} F_{c',p} & \text{if } p \in \{p_1, \ldots, p_{m_1}\} \\ & \quad \text{and } c' = c_1 \\ F_{c',p} & \text{if } p \in \{p_{m_1+1}, \ldots, p_{m_2}\} \\ & \quad \text{and } c' = c_2 \\ F_{c',p} & \text{if } p \in \{p_{m_2+1}, \ldots, p_n\} \\ & \quad \text{and } (c' \neq c_1 \vee p \notin P_{d_1}) \\ & \quad \text{and } (c' \neq c_2 \vee p \notin P_{d_2}) \\ 0 & \text{otherwise.} \end{cases} \tag{53}$$

If p is one of the first m_1 periods, the capacity is set to $F_{c',p}$ if $c' = c_1$; if p is one of the next $m_2 - m_1$ periods, the capacity is set to $F_{c',p}$ if $c' = c_2$; if p is one of the $n - m_2$ last periods, the capacity is set to $F_{c',p}$ if p

does not belong to day d_1 or if $c' \neq c_1$ and if p does not belong to d_2 or $c' \neq c_2$. A maximum flow problem is then solved and if the total amount of flow is less than L_γ, then the two patterns cannot both be selected in a feasible solution and an edge in the conflict graph is added between $v^{k_1}_{c_1,d_2}$ and $v^{k_2}_{c_2,d_2}$.

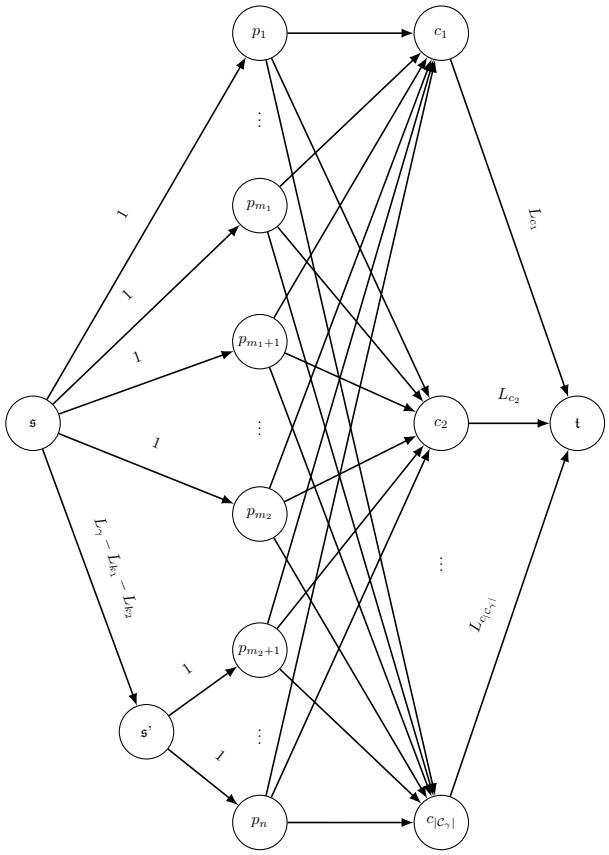

Figure 4: Given a clique $\gamma \in \Gamma$, maximum flow graph to identify if patterns $k_1 \in \mathcal{K}_{c_1,d_1}$ and $k_2 \in \mathcal{K}_{c_2,d_2}$ for some courses $c_1, c_2 \in \mathcal{C}_\gamma$ and days $d_1, d_2 \in \mathcal{D}$ cannot both be contained in a feasible solution.

Even if the maximum flow problem can be solved efficiently, we need to solve many problems, resulting in a long total running time. This can be speeded up by, instead of creating a graph for each pair of nodes in the conflict graph, we create the graph for an entire clique and then adjust the capacities according to the courses and patterns that are under consideration. Consider a clique $\gamma \in \Gamma$ and create the same nodes as above, i.e., the source node $\mathfrak{s}$, the dummy source node $\mathfrak{s}$', the sink $\mathfrak{t}$, a node for each period and a node for each course. As before, each node representing a course $c \in \mathcal{C}_\gamma$ has an outgoing arc connected to the sink $\mathfrak{t}$ with a capacity of L_c. There is an arc from the source $\mathfrak{s}$ to the dummy source $\mathfrak{s}$', but this time the capacity is initially set to zero. Each period node has an incoming arc from both the source $\mathfrak{s}$ and the dummy source $\mathfrak{s}$' and has an outgoing arc to each course $c \in \mathcal{C}_\gamma$ that can be feasibly scheduled in that period. The capacities of all the arcs entering and leaving the periods are initially set to zero. All capacities that are initially set to zero are adjusted according to the courses, days and patterns that are under consideration. At first the flow on all arcs is zero. This flow is a maximum integer flow since all arcs leaving the source have zero capacity, no capacities are violated and all node balancing constraints are satisfied. After this, the approach is then to run an iterative algorithm that maintains a maximum integer flow as the loop invariant. Each iteration consists of three steps that first picks two patterns to investigate and adjust the capacities accordingly, possibly creating an infeasible flow. The second step is to repair the possibly infeasible flow created by the adjustment of the capacities. The last step is to recalculate the maximum flow.

Step 1 – Select patterns and adjust capacities. Select courses $c_1, c_2 \in \mathcal{C}_\gamma$, days $d_1, d_2 \in \mathcal{D}$ and patterns $k_1 \in \mathcal{K}_{c_1,d_1}$ and $k_2 \in \mathcal{K}_{c_2,d_2}$ which have not been considered yet and where there is no edge between $v^{k_1}_{c_1,d_1}$ and $v^{k_2}_{c_2,d_2}$ in the pattern conflict graph. The capacities are then adjusted according to the selected patterns. First the capacity on the arc from the source $\mathfrak{s}$ to the dummy $\mathfrak{s}$' is set to the total amount of lectures that need to be scheduled for the clique minus the amount of lectures in the two patterns:

$$a\left(\mathfrak{s}, \mathfrak{s}'\right) = L_\gamma - L_{k_1} - L_{k_2}. \tag{54}$$

The capacities from the source $\mathfrak{s}$ to a node for some period p are set to one if the period is part of any of the two patterns, otherwise it is set to zero:

$$a\left(\mathfrak{s}, p = (d,t)\right) = \begin{cases} 1 & \text{if } d = d_1 \wedge a^{k_1}_t = 1 \\ 1 & \text{if } d = d_2 \wedge a^{k_2}_t = 1 \\ 0 & \text{otherwise.} \end{cases} \tag{55}$$

The capacities from $\mathfrak{s}$' to a node for some period p are the opposite of the arcs from the source $\mathfrak{s}$:

$$a\left(\mathfrak{s}', p = (d,t)\right) = \begin{cases} 0 & \text{if } d = d_1 \wedge a^{k_1}_t = 1 \\ 0 & \text{if } d = d_2 \wedge a^{k_2}_t = 1 \\ 1 & \text{otherwise.} \end{cases} \tag{56}$$

Capacities (54), (55) and (56) together ensure that the maximum flow can only be equal to the total number of lectures if some flow is sent through all the periods from the selected patterns. One must also ensure that the flow leaving the periods from the selected patterns are only sent to the designated courses, i.e., for each period $p = (d,t)$, if $d = d_1 \wedge a^{k_1}_t = 1$, the flow can only be sent to c_1 and if $d = d_2 \wedge a^{k_2}_t = 1$, the flow can only be sent to c_2. Furthermore, since one pattern must be selected for each course and each day, then c_1 cannot be scheduled in any period at day d_1 which is not contained in pattern k_1; the same holds true for course c_2, day d_2 and pattern k_2. This results in the following capacities:

$$a\left(p = (d,t), c\right) = \begin{cases} F_{c,p}\, a^{k_1}_t & \text{if } d = d_1 \wedge c = c_1 \\ F_{c,p}\, a^{k_2}_t & \text{if } d = d_2 \wedge c = c_2 \\ F_{c,p}\left(1 - a^{k_1}_t\right) & \text{if } d = d_1 \wedge c \neq c_1 \\ F_{c,p}\left(1 - a^{k_2}_t\right) & \text{if } d = d_2 \wedge c \neq c_2 \\ F_{c,p} & \text{otherwise.} \end{cases} \tag{57}$$

Step 2 – Repair flow. If the capacity adjustment from Step 1 is making the flow infeasible, then the flow is adjusted to regain feasibility. Let $f(u,v)$ be the flow on arc (u,v) and f_{viol} be the sum of all the excess flow on the arcs:

$$f_{\text{viol}} = \sum_{(u,v)} \left(f(u,v) - a(u,v)\right)^+.$$

The adjustments are done by going through all the arcs in the flow graph. If the capacity is violated on an arc (u,v), it must be because the amount of flow is at least one unit more than the capacity: $f(u,v) \geq a(u,v)+1$. This is because we know from the loop invariant that the flow is integral and because the capacities are either changed from one to zero or from zero to one. This means that there must exist some directed path going from the source $\mathfrak{s}$ to the sink $\mathfrak{t}$ through arc (u,v) with a flow of at least one unit on all its arcs. Such a path is illustrated in Figure 5 and can easily be found by identifying two shortest paths (with respect to the number of arcs), from $\mathfrak{s}$ to u and from v to $\mathfrak{t}$ using only arcs where there are at least one unit of flow. These paths are then put together into one path with the arc (u,v). Let $\Delta^{\mathfrak{s},\mathfrak{t}}_{(u,v)} \geq 1$ be the smallest excess flow on that path, we then decrease the flow on this path by $\Delta^{\mathfrak{s},\mathfrak{t}}_{(u,v)}$. This reduces f_{viol} and we can just repeat this process until f_{viol} is equal to zero. Since the flow is initially integral, $\Delta^{\mathfrak{s},\mathfrak{t}}_{(u,v)}$ is always integer and the repaired flow remains integral.

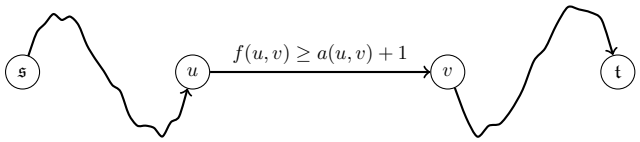

Figure 5: Illustration of a path with at least one unit flow from s to t through an arc (u, v) where the capacity is violated.

Step 3 – Maximum flow. The last step is to run a maximum flow algorithm initialized with the possibly repaired flow. Any augmenting path algorithm can be used. For our implementation we used the algorithm described by Edmonds and Karp (1972).

5 Computational results

We have compared our results to the best ones that can be found in the literature. For the ITC2007 competition, a benchmarking tool was provided. It returns the number of seconds that the competitors were allowed to run their algorithms on their own computers, restricted to one core. This number of seconds is referred to as one CPU unit. Our tests are conducted on an Intel® Core™ I5-3570K 3.4GHz CPU with 16GB RAM running Windows 10. The benchmark tool returned 208 seconds as one CPU unit for our computer. The ILP solver we used is from Gurobi Optimization Inc. (2015) version 6.5.

Since our mixed-integer programming (MIP) algorithm provides lower bounds, we are only comparing our algorithm with other MIP-based algorithms from the literature that obtain lower bounds. The algorithms we compare are those identified by LL12 (Lach and Lübbecke, 2012), BMPR10 (Burke et al, 2010), HB11 (Hao and Benlic, 2011), CCRT13 (Cacchiani et al, 2013) and Patterns (this paper). We have run the tests with time limits of 1, 10 and 40 CPU units as it has also been done for the other algorithms. The running time of the preprocessing and the generation of the valid inequalities is included in the total time spent by our algorithm. 21 data sets have been provided for the competition, however only 14 of these were available for some of the algorithms. All data sets can be obtained from Bonutti et al (2016).

Table 2 presents results for comparing these different algorithms on the first fourteen data sets. For each algorithm, we provide the lower bound obtained (lb) and the relative distance ($\%gap$) to the best known upper bound (ub) calculated as $(ub - lb)/ub$. The lower bounds in bold font are best lower bounds. Those underlined are obtained by a single algorithm.

Line Avg. $\%gap$ specifies the average percentage gap obtained for all the fourteen instances. The two last lines report the number of times an algorithm obtains a best lower bound (Best lb) and a better one than the others (Better lb).

In Table 2(a), it can be seen that the pattern formulation yields a best lower bound for 11 of the 14 data sets. The three exceptions are for data sets comp01, comp09 and comp11. For data set comp01, we do not achieve the best lower bound because we do not consider the room penalties whereas the others do. For data set comp11, the number of time slots per day is 9, resulting in a potential of 512 patterns for each course and each day. In this case, our algorithm reaches the time limit of 1 CPU unit before completing the preprocessing and the valid inequality generation. This is also the case when we look at Table 2(b). In Table 2(c), it can be seen that our algorithm is the best for 13 instances, only producing a worse bound again for data set comp01. In half of the instances, the pattern formulation provides a better bound. Furthermore, for instance comp12, the lower 165 are an improvement over the currently best known bound 138.

Because Cacchiani et al (2013) also report results on the last seven instances for a time limit of 40 CPU units, we also compare our algorithm to their algorithm on all data sets, except comp11. Indeed, an optimal solution of this data set has a zero cost and therefore Cacchiani et al (2013) deemed this uninteresting as a trivial lower bound is also zero. The results are presented in Table 3 where it can be seen that the pattern formulation provides the best lower bound 19 times out of 20. We only get a worse bound for data set comp01 because we do not consider any room related penalties. Finally, in half of these data sets, a better lower

Table 2: Lower bounds for various algorithms given 1, 10, and 40 CPU time units.

(a) Time limit: One CPU unit

Instance		LL12		BMPR10		HB11		CCRT13		Patterns	
	ub	*lb*	*%gap*	*lb*	*%gap*	*lb*	*%gap*	*lb*	*%gap*	*lb*	*%gap*
comp01	5	4	20.0	0	100.0	4	20.0	**5**	0.0	0	100.0
comp02	24	0	100.0	0	100.0	**10**	58.3	0	100.0	**10**	58.3
comp03	64	0	100.0	25	60.9	26	59.4	24	62.5	**41**	35.9
comp04	35	22	37.1	**35**	0.0	**35**	0.0	**35**	0.0	**35**	0.0
comp05	284	92	67.6	119	58.1	19	93.3	6	97.9	**154**	45.8
comp06	27	7	74.1	13	51.9	12	55.6	0	100.0	**19**	29.6
comp07	6	0	100.0	**6**	0.0	5	16.7	0	100.0	**6**	0.0
comp08	37	30	18.9	**37**	0.0	**37**	0.0	**37**	0.0	**37**	0.0
comp09	96	37	61.5	68	29.2	39	59.4	**92**	4.2	82	14.6
comp10	4	2	50.0	3	25.0	**4**	0.0	0	100.0	**4**	0.0
comp11	0	**0**	0.0	**0**	0.0	**0**	0.0	-	-	-	-
comp12	298	29	90.3	101	66.1	43	85.6	0	100.0	**109**	63.4
comp13	59	33	44.1	52	11.9	46	22.0	57	3.4	**59**	0.0
comp14	51	40	21.6	41	19.6	41	19.6	32	37.3	**45**	11.8
Avg. *%gap*			56.5		37.3		35.0		50.4		27.9
Best *lb*		**1**		**4**		**5**		**4**		**11**	
Better *lb*								**2**		**6**	

(b) Time limit: 10 CPU units

Instance		LL12		BMPR10		HB11		CCRT13		Patterns	
	ub	*lb*	*%gap*	*lb*	*%gap*	*lb*	*%gap*	*lb*	*%gap*	*lb*	*%gap*
comp01	5	4	20.0	4	20.0	4	20.0	**5**	0.0	0	100.0
comp02	24	8	66.7	0	100.0	12	50.0	**16**	33.3	14	41.7
comp03	64	0	100.0	33	48.4	34	46.9	**52**	18.8	46	28.1
comp04	35	28	20.0	**35**	0.0	**35**	0.0	**35**	0.0	**35**	0.0
comp05	284	25	91.2	111	60.9	69	75.7	6	97.9	**178**	37.3
comp06	27	10	63.0	15	44.4	12	55.6	11	59.3	**20**	25.9
comp07	6	2	66.7	**6**	0.0	**6**	0.0	**6**	0.0	**6**	0.0
comp08	37	34	8.1	**37**	0.0	**37**	0.0	**37**	0.0	**37**	0.0
comp09	96	41	57.3	65	32.3	67	30.2	92	4.2	**94**	2.1
comp10	4	**4**	0.0	**4**	0.0	**4**	0.0	2	50.0	**4**	0.0
comp11	0	**0**	0.0	**0**	0.0	**0**	0.0	-	-	-	-
comp12	298	32	89.3	95	68.1	78	73.8	0	100.0	**159**	46.6
comp13	59	39	33.9	52	11.9	53	10.2	57	3.4	**59**	0.0
comp14	51	41	19.6	42	17.6	43	15.7	48	5.9	**51**	0.0
Avg. *%gap*			45.6		28.8		27.0		26.6		21.7
Best *lb*		**2**		**5**		**5**		**6**		**10**	
Better *lb*								**3**		**6**	

(c) Time limit: 40 CPU units

Instance		LL12		BMPR10		HB11		CCRT13		Patterns	
	ub	*lb*	*%gap*	*lb*	*%gap*	*lb*	*%gap*	*lb*	*%gap*	*lb*	*%gap*
comp01	5	4	20.0	**5**	0.0	4	20.0	**5**	0.0	0	100.0
comp02	24	11	54.2	1	95.8	12	50.0	16	33.3	**20**	16.7
comp03	64	25	60.9	33	48.4	36	43.8	**52**	18.8	**52**	18.8
comp04	35	28	20.0	**35**	0.0	**35**	0.0	**35**	0.0	**35**	0.0
comp05	284	108	62.0	114	59.9	80	71.8	166	41.5	**191**	32.7
comp06	27	10	63.0	16	40.7	16	40.7	11	59.3	**24**	11.1
comp07	6	**6**	0.0	**6**	0.0	**6**	0.0	**6**	0.0	**6**	0.0
comp08	37	**37**	0.0	**37**	0.0	**37**	0.0	**37**	0.0	**37**	0.0
comp09	96	46	52.1	66	31.3	67	30.2	92	4.2	**96**	0.0
comp10	4	**4**	0.0	**4**	0.0	**4**	0.0	2	50.0	**4**	0.0
comp11	0	**0**	0.0	**0**	0.0	**0**	0.0	-	-	**0**	0.0
comp12	298	53	82.2	95	68.1	84	71.8	100	66.4	**165**	44.6
comp13	59	41	30.5	54	8.5	55	6.8	57	3.4	**59**	0.0
comp14	51	46	9.8	42	17.6	43	15.7	48	5.9	**51**	0.0
Avg. *%gap*			32.8		26.5		25.1		20.2		16.0
Best *lb*		**4**		**6**		**5**		**6**		**13**	
Better *lb*										**7**	

bound is obtained by our algorithm compared to CCRT13. The last comparison is against the best known lower bounds as reported on the website of Bonutti et al (2016). The algorithms or the computational times used for these bounds are not always clear, so we have chosen to run our algorithm with a time limit of 100

Table 3: Cacchiani et al (2013) vs. our algorithm, given 40 CPU units.

Instance		CCRT13		Patterns	
	ub	*lb*	*%gap*	*lb*	*%gap*
comp01	5	**5**	0.0	0	100.0
comp02	24	16	33.3	**20**	16.7
comp03	64	**52**	18.8	**52**	18.8
comp04	35	**35**	0.0	**35**	0.0
comp05	284	166	41.5	**191**	32.7
comp06	27	11	59.3	**24**	11.1
comp07	6	**6**	0.0	**6**	0.0
comp08	37	**37**	0.0	**37**	0.0
comp09	96	92	4.2	**96**	0.0
comp10	4	2	50.0	**4**	0.0
comp12	298	100	66.4	**165**	44.6
comp13	59	57	3.4	**59**	0.0
comp14	51	48	5.9	**51**	0.0
comp15	62	**52**	16.1	**52**	16.1
comp16	18	13	27.8	**18**	0.0
comp17	56	48	14.3	**52**	7.1
comp18	61	**52**	14.8	**52**	14.8
comp19	57	48	15.8	**57**	0.0
comp20	4	**4**	0.0	**4**	0.0
comp21	74	**68**	8.1	**68**	8.1
Avg. *%gap*			18.1		12.9
Best *lb*		**9**		**19**	
Better *lb*		**1**		**10**	

CPU units to see if we can improve some of the bounds. The results are presented in Table 4 where it can be seen that for comp03, comp12 and comp15, new lower bounds are obtained. The pattern formulation is also able to match most of the currently best known bounds, that is, 16 times out of 21. For the data sets where it produces worse bounds, it can be seen that it is still very close except for data set comp18. The reason for this is hard to identify as it is not known what was the time limit when this bound was obtained.

Table 4: Comparison with the best known bounds given 100 CPU units.

Instance		Best known		Patterns	
	ub	*lb*	*%gap*	*lb*	*%gap*
comp01	5	5	0.0	0	100.0
comp02	24	24	0.0	**24**	0.0
comp03	64	52	18.8	**54**	15.6
comp04	35	35	0.0	**35**	0.0
comp05	284	211	25.7	210	26.1
comp06	27	27	0.0	26	3.7
comp07	6	6	0.0	**6**	0.0
comp08	37	37	0.0	**37**	0.0
comp09	96	96	0.0	**96**	0.0
comp10	4	4	0.0	**4**	0.0
comp11	0	0	0.0	**0**	0.0
comp12	298	138	53.7	**175**	41.3
comp13	59	59	0.0	**59**	0.0
comp14	51	51	0.0	**51**	0.0
comp15	62	52	16.1	**54**	12.9
comp16	18	18	0.0	**18**	0.0
comp17	56	56	0.0	53	5.4
comp18	61	61	0.0	52	14.8
comp19	57	57	0.0	**57**	0.0
comp20	4	4	0.0	**4**	0.0
comp21	74	74	0.0	**74**	0.0
Avg. *%gap*			5.4		10.5
Best *lb*				**16**	
Better *lb*				**3**	

6 Perspective

The pattern formulation improves three of the lower bounds for the ITC2007 data sets. It also has some other benefits such as the possibility of including specialized constraints to the daily time patterns. For example, if a course has two lectures in a given day, these should not be too far apart, or if a course has any lectures in a day, there should be a minimum or maximum number of lectures scheduled that day. These constraints could either be hard or soft and it can easily be checked whether the patterns are violating them. Such constraints can even be included non-linearly, e.g. if the distance between two lectures for some courses scheduled on the same day is defined as the number of empty time slots between them, then such distance could be penalized by squaring the number or by some other non-linear expression.

The pattern formulation could be extended to consider an entire week instead of only one day at a time. This however results in many more patterns, which might require the implementation of a Branch-and-Price algorithm. One drawback is that we loose the benefit of the cutting planes already available in a commercial code such as Gurobi. A shift to a Branch-and-Price framework such as SCIP (Achterberg, 2009) could be a way to get around this issue.

In this paper, we describe how a conflict graph on the patterns is constructed. This is used to statically generate clique cuts for the pattern formulation. It could be interesting to find out if more edges can be added to the graph. Furthermore instead of adding the cuts statically, another approach could be to use the graph to generate the clique cuts dynamically as in a Branch-and-Cut algorithm.

References

Achterberg T (2009) SCIP: Solving constraint integer programs. Mathematical Programming Computation 1(1):1–41, http://mpc.zib.de/index.php/MPC/article/view/4

Asín Achá R, Nieuwenhuis R (2012) Curriculum-based course timetabling with sat and maxsat. Annals of Operations Research pp 1–21

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

Bettinelli A, Cacchiani V, Roberti R, Toth P (2015) An overview of curriculum-based course timetabling. TOP 1–37

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, Gaspero LD, Schaerf A (2016) Curriculum-based course timetabling. http://tabu.diegm.uniud.it/ctt/index.php[Retrieved 19/4-2016]

Bron C, Kerbosch J (1973) Algorithm 457: Finding all cliques of an undirected graph. Commun ACM 16(9):575–577, DOI 10.1145/362342.362367, http://doi.acm.org/10.1145/362342.362367

Burke EK, Mareček J, Parkes AJ, Rudová H (2008) Penalising patterns in timetables: Novel integer programming formulations. In: Kalcsics J, Nickel S (eds) Operations Research Proceedings 2007, Operations Research Proceedings, vol 2007, Springer Berlin Heidelberg, 409–414, 10.1007/978-3-540-77903-2_63

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

Burke EK, Mareček J, Parkes AJ, Rudová H (2012) A branch-and-cut procedure for the udine course timetabling problem. Annals of Operations Research 194(1):71–87

Cacchiani V, Caprara A, Roberti R, Toth P (2013) A new lower bound for curriculum-based course timetabling. Computers & Operations Research 40(10):2466–2477

Di Gaspero L, McCollum B, Schaerf A (2007) The second international timetabling competition (ITC-2007): Curriculum-based course timetabling (track 3). Tech. rep., School of Electronics, Electrical Engineering and Computer Science, Queen's University SARC Building, Belfast, United Kingdom

Edmonds J, Karp RM (1972) Theoretical improvements in algorithmic efficiency for network flow problems. J ACM 19(2):248–264, DOI 10.1145/321694.321699, http://doi.acm.org.globalproxy.cvt.dk/10.1145/321694.321699

Glover F, Laguna M (2013) Handbook of Combinatorial Optimization, Springer New York, New York, NY, chap Tabu Search, 3261–3362. DOI 10.1007/978-1-4419-7997-1_17, http://dx.doi.org/10.1007/978-1-4419-7997-1_17

Gurobi Optimization Inc (2015) Gurobi optimizer reference manual. http://www.gurobi.com

Hao JK, Benlic U (2011) Lower bounds for the ITC-2007 curriculum-based course timetabling problem. European Journal of Operational Research 212(3):464–472

Kou LT, Stockmeyer LJ, Wong CK (1978) Covering edges by cliques with regard to keyword conflicts and intersection graphs. Communications of the ACM 21(2):135–139

Lach G, Lübbecke M (2008) Optimal university course timetables and the partial transversal polytope. In: McGeoch C (ed) Experimental Algorithms, Lecture Notes in Computer Science, vol 5038, Springer Berlin / Heidelberg, 235–248

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

McCollum B, Schaerf A, Paechter B, McMullan P, Lewis R, Parkes AJ, Gaspero LD, Qu R, Burke EK (2010) Setting the research agenda in automated timetabling: The second international timetabling competition. INFORMS Journal on Computing 22(1):120–130

Van Roy TJ, Wolsey LA (1987) Solving mixed integer programming problems using automatic reformulation. Operations Research 35(1):45–57