Benders' Decomposition for Curriculum-Based Course Timetabling

Bagger, Niels-Christian F.; Sørensen, Matias; Stidsen, Thomas R.

Published in:
Computers and Operations Research

Link to article, DOI:
10.1016/j.cor.2017.10.009

Publication date:
2018

Document Version
Peer reviewed version

Benders' Decomposition for Curriculum-Based Course Timetabling

Niels-Christian F. Bagger[a,b,1,*], Matias Sørensen[a,b], Thomas R. Stidsen[a]

[a] *mORetime research group, Management Science, Department of Management Engineering, Technical University of Denmark, Produktionstorvet, Building 426B, DK-2800 Kgs. Lyngby, Denmark, http://www.moretime.man.dtu.dk*
[b] *MaCom A/S, Vesterbrogade 48, 1., DK-1620 København V, Denmark*

Abstract

In this paper we applied Benders' decomposition to the Curriculum-Based Course Timetabling (CBCT) problem. The objective of the CBCT problem is to assign a set of lectures to time slots and rooms. Our approach was based on segmenting the problem into time scheduling and room allocation problems. The Benders' algorithm was then employed to generate cuts that connected the time schedule and room allocation. We generated only feasibility cuts, meaning that most of the solutions we obtained from a mixed integer programming solver were infeasible, therefore, we also provided a heuristic in order to regain feasibility.

We compared our algorithm with other approaches from the literature for a total of 32 data instances. We obtained a lower bound on 23 of the instances, which were at least as good as the lower bounds obtained by the state-of-the-art, and on eight of these, our lower bounds were higher. On two of the instances, our lower bound was an improvement of the currently best-known. Lastly, we compared our decomposition to the model without the decomposition on an additional six instances, which are much larger than the other 32. To our knowledge, this was the first time that lower bounds were calculated for these six instances.

Keywords: University Course Timetabling, Integer Programming, Benders' Decomposition, Maximum Flow, Minimum Cut

1. Curriculum-based Course Timetabling

In this work we considered the Curriculum-Based Course Timetabling Problem (CBCT) introduced in Track 3 of the Second International Timetabling Competition (ITC2007) as described by Di Gaspero et al. (2007), McCollum et al. (2010) and Bonutti et al. (2012). Most of the work on CBCT focused on the discovery of high-quality solutions using heuristics. The drawback of these heuristics is that they do not provide any proof of quality (e.g., how far from optimality the solutions actually are). We need bounding and exact methods to be able to validate the quality of the heuristics and not much work has been put into the development of these methods. In this article we applied Benders' decomposition to a Mixed Integer Programming (MIP) model that we presented in Bagger et al. (2016). We submitted the technical report (Bagger et al., 2016) to the Annals of Operations Research (ANOR). ANOR is, as yet, unaware of our work in this article as we had not fully developed the method. We provide the model from the report in section 2.1. The proof of the correctness of the model is in the technical report which is essential, as the application of the decomposition in this paper relies on that model.

The CBCT problem entails that we must schedule weekly lectures for multiple courses into time periods and assign the lectures to rooms. We are given a set of days, each divided into a set of time slots. We refer to a day and time slot combination as a period. The basic entities of the problem are the *courses* to schedule, the *periods*, and the *rooms* that are available. The problem originates from a real world application and has thus received significant attention since the competition. Each course contains a number of lectures that must all be scheduled within a period, and assigned a room. Furthermore, all of the lectures are to be scheduled within distinct periods. This requirement is referred to as the Lectures (**L**) constraint. For a course, some of the periods can be specified as *unavailable*, i.e., periods where it is not allowed to schedule the course. This requirement is referred to as the Availability (**A**) constraint. There are no constraints on assigning the courses to rooms, i.e., any course can be assigned to any room. Aside from the courses, the periods and the rooms, the problem also contains *lecturers* and *curricula*; hence the name *Curriculum-Based Course Timetabling*. Each course is taught by a lecturer, and a curriculum is a set of courses that may be followed by the same students. If two courses are taught by the same lecturer, or belong to the same curriculum they cannot have lectures scheduled within the same periods. This requirement is referred to as

*Corresponding author
Email addresses: nbag@dtu.dk (Niels-Christian F. Bagger), ms@macom.dk (Matias Sørensen), sorensen.matias@gmail.com (Matias Sørensen), thst@dtu.dk (Thomas R. Stidsen)
[1] ORCiD: 0000-0003-4665-6761

the Conflicts (**C**) constraint. For each room, one lecture, at most, can be assigned in any period, which is referred to as the Room Occupancy (**RO**) constraint. The objective of the CBCT is to develop a timetable that fulfills all of the latter mentioned requirements, **L**, **A**, **C** and **RO**, while minimizing a weighted sum of the violation of four soft constraints; Room Capacity (**RC**), Room Stability (**RStab**), Minimum Working Days (**MWD**) and Isolated Lectures (**IL**). When a course is assigned to a room where the number of seats is smaller than the number of students that are attending the course, then the constraint **RC** is violated by one for each student above the capacity of the room. It is desirable to assign lectures from the same course to as few distinct rooms as possible. A course violates the constraint **RStab** by the total number of distinct rooms that it is assigned to minus one. The constraint **MWD** is the desire to spread the lectures across a given number of days. We say that a day is a working day for a course if at least one lecture from the course is scheduled in a time slot on that day. For each course, a number of minimum working days is provided and if the number of working days is below this number then the violation of the constraint is the difference. The final constraint **IL** is associated with the curricula. For each curriculum it is desired to have as few *isolated* lectures as possible. Each *isolated* lecture counts as one violation. A curriculum has an *isolated* lecture in a period if any of the courses belonging to the curriculum has a lecture scheduled in the period, and none of the courses have lectures scheduled in the *adjacent* periods. We say that two periods are adjacent if they belong to the same day and are in consecutive time slots.

In the following section 1.1 we provide an overview of other approaches that were applied to CBCT. In section 2 we initially provide a brief introduction to Benders' decomposition, and then describe how we applied it to CBCT. In section 3 we describe a heuristic to repair partially infeasible solutions, where by partially infeasible we refer to solutions wherein the time schedule is feasible, but not the room assignment. In section 4 we describe the computational results. Lastly, in section 5 we state our conclusions on this work.

1.1. Related Research

As we considered a MIP model for the problem, we focused primarily on other MIP-based approaches in the literature. For a thorough overview of the problem and different approaches for CBCT we refer to Bettinelli et al. (2015).

Burke et al. (2008, 2010) introduced a compact MIP formulation that was exact, in the sense that the optimal solution can be found by a generic MIP solver given enough computational resources. However, many instances of the CBCT could not be solved for this formulation within a reasonable time using a MIP solver; hence Burke et al. (2010) proposed methods to derive lower and upper bounds. They obtained lower bounds by aggregating the rooms into *multi-rooms*. For each *multi-room* the number of lectures that can be scheduled in it, for any period, is equal to the number of rooms that were aggregated. This problem provides a lower bound for CBCT. To obtain an upper bound they fixed the periods (or portions thereof) according to the solution from the lower bounding mechanism, and subsequently assigned rooms to the lectures. Burke et al. (2012) proposed an exact branch-and-cut algorithm which they also based on the compact formulation; however, some of the objective costs were left out and instead added as cuts during the solution process. This can be seen as a Benders' decomposition; howeber, rather than generating the cuts dynamically, they were generated a priori and then added as required.

Lach and Lübbecke (2008, 2012) proposed a method that divided the CBCT into two stages. The began by grouping the rooms together such that if two rooms had the same capacity, then they were in the same group. Then, in the first stage, they scheduled the courses into periods and assigned them to these capacities. This method is a Benders' Decomposition (Lübbecke, 2015). In the second stage, they assigned the courses to the rooms, where the solution from the first stage was employed to fix the courses for the determined periods and the selected room capacities.

Hao and Benlic (2011) divided the MIP model that Lach and Lübbecke (2012) used in the first-stage into smaller components by relaxing or removing some of the constraints. These relaxations made it possible to decompose the model into a set of sub-problems, where they calculated a lower bound for each sub-problem. The sum of all these lower bounds was then a lower bound for CBCT.

Cacchiani et al. (2013) presented multiple extended MIP formulation, i.e., models with an exponential number of variables. The approach that provided the best results divided the problem into two parts; one that focused on the time scheduling-related soft constraints, while the other focused on the room-related soft constraints. They calculated a lower bound for each part and the sum of these lower bounds was then a lower bound for CBCT.

In Bagger et al. (2016) two MIP models were presented that were inspired by Lach and Lübbecke (2008, 2012) and Burke et al. (2008, 2010). The division of the problem described by Lach and Lübbecke (2012) was applied (excluding the notion of distinct capacities), where it was shown that the two stages could be connected using two underlying flow network formulations. The first formulation was based on a minimum cost flow network, and performed as the best of the two. The second formulation was based on a multi-commodity flow problem which was the formulation where we applied Benders' decomposition (Benders, 1962) in this article. The rationale for using the last formulation, although it did not perform as well as the first, is that the underlying network is a feasibility problem. Therefore, we did not need to generate any optimality cuts, only feasibility cuts.

2. Benders' Decomposition

In this section we give an introduction to Benders' decomposition followed by our application of the technique. Our introduction is a crude overview, and we refer to (Benders, 1962) and (Martin, 1999, chapter 10) for a detailed description. We describe the method based on a model that contains two types of variables, x and y. The x variables are non-negative continuous variables, and we do not have any assumptions on the y variables, i.e., $x \geq 0$ and $y \in Y$ where Y can be any domain (e.g., the set of integers). Consider the MIP model (1).

$$\begin{aligned} \min \quad & c^\top x + f(y) \\ \text{s.t.} \quad & Ax + B(y) \geq b \\ & y \in Y \\ & x \geq 0 \end{aligned} \tag{1}$$

In the model (1) $c \in \mathbb{R}^n$ is the cost vector of the x variables, $A \in \mathbb{R}^{n \times m}$ is the constraint matrix of the x variables and $b \in \mathbb{R}^m$ is the right-hand-side vector of the constraints. $f : Y \to \mathbb{R}$ is some function to evaluate the cost of the y variables and B is a vector function that evaluates the contribution of the y variables for the constraints. If we fix the y variables to some value in the domain Y then what remains is a linear programme (LP). This assumption can be extended as described by Geoffrion (1972), but we stick to the (LP) case in this context. Model (1) can be rewritten to model (2).

$$\begin{aligned} \min \quad & f(y) + z \\ \text{s.t.} \quad & z \geq \min_{x \geq 0} \left\{ c^\top x \mid Ax \geq b - B(y) \right\} \\ & y \in Y \\ & z \in \mathbb{R} \end{aligned} \tag{2}$$

In the model (2) there is an inner optimization problem in the constraints. If the y variables are fixed, then this is an LP and we can change it into its dual LP as in model (3).

$$\begin{aligned} \min \quad & f(y) + z \\ \text{s.t.} \quad & z \geq \max_{\pi \geq 0} \left\{ (b - B(y))\, \pi \mid A^\top \pi \leq c \right\} \\ & y \in Y \\ & z \in \mathbb{R} \end{aligned} \tag{3}$$

One interesting aspect of the inner optimization problem in model (3) is that the corresponding polytope is independent of the values of the y variables. Hence, if the polytope $\left\{ A^T \pi \leq c \right\}$ is non-empty, then we can reformulate the problem using the extreme points Π^p and extreme rays Π^r as in model (4).

$$\begin{aligned} \min \quad & f(y) + z \\ \text{s.t.} \quad & z \geq (b - B(y))\, \overline{\pi}^p \quad \forall \overline{\pi}^p \in \Pi^p \\ & 0 \geq (b - B(y))\, \overline{\pi}^r \quad \forall \overline{\pi}^r \in \Pi^r \\ & y \in Y \\ & z \in \mathbb{R} \end{aligned} \tag{4}$$

Model (4) is referred to as Benders' master problem. For a given solution $\overline{y}$ model (5) is referred to as Benders' subproblem.

$$\begin{aligned} \max \quad & (b - B(\overline{y}))\, \pi \\ \text{s.t.} \quad & A^\top \pi \leq c \\ & \pi \geq 0 \end{aligned} \tag{5}$$

As the number of extreme points and rays can be exponentially large, a strategy for solving the model is to relax the master problem by removing some (or all) of the constraints that originates from the extreme points and rays and then iteratively add them as required. This is done by finding a solution $\overline{y}$ for the master problem (4) and inserting the solution into the subproblem (5). The subproblem is then solved to obtain an extreme point $\overline{\pi}^p$ or ray $\overline{\pi}^r$ and the corresponding cut is added to the master problem if it is violated. This is done iteratively as illustrated in Figure 1.

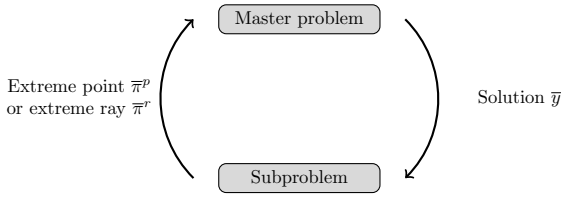

Figure 1: The iterative loop of Benders' algorithm.

A lower bound on the (relaxed) master problem is a lower bound on the original model, and if the subproblem returns an extreme point, then this provides an upper bound. The iterative process is run until some stopping criterion is met (e.g., a time limit is reached, or the lower and upper bounds are sufficiently close).

In the following section 2.1 we present the model we employed for the decomposition and Benders' master problem. In section 2.2 we state the sub-problems we used to generate Benders' cuts. For all of the models we are given the set of courses $\mathcal{C}$, the set of periods $\mathcal{P}$ and the set of rooms $\mathcal{R}$. We are also provided with the set of days $\mathcal{D}$ and the set of periods $\mathcal{P}_d$ belonging to each day $d \in \mathcal{D}$. For each period $p \in \mathcal{P}$ the set θ_p is the set of periods that are adjacent to p, i.e., the periods that belong to the same day and are in consecutive time slots. Furthermore, we have the set of lecturers $\mathcal{L}$ and curricula $\mathcal{Q}$. For each curriculum $q \in \mathcal{Q}$ we have the set of courses $\mathcal{C}_q$ that belongs to the curriculum. We also define the set Γ which is the set of course-cliques. A course-clique $\gamma \in \Gamma$ is a set of courses

$\mathcal{C}_\gamma$ such that for each period $p \in \mathcal{P}$ one lecture, at most, from $\mathcal{C}_\gamma$ can be scheduled in that period. We found the set of course-cliques in the same way as Bagger et al. (2016). For each course $c \in \mathcal{C}$ we have the number of lectures that is required to be scheduled $L_c \in \mathbb{Z}^+$, where $\mathbb{Z}^+$ is the set of non-negative integers, and the minimum number of requested working days $M_c \in \mathbb{Z}^+$. We also have the number of students attending the course $S_c \in \mathbb{Z}^+$, and for each period $p \in \mathcal{P}$, the parameter $F_{c,p} \in \mathbb{B}$ that specifies whether or not it is feasible to schedule c in p. For each room, we have the parameter $C_r \in \mathbb{Z}^+$, which is the capacity of the room. We have $W^{\mathbf{MWD}} \in \mathbb{R}^+$, $W^{\mathbf{IL}} \in \mathbb{R}^+$, $W^{\mathbf{RC}} \in \mathbb{R}^+$ and $W^{\mathbf{RStab}} \in \mathbb{R}^+$, which are the non-negative weights of the soft constraints, **MWD**, **IL**, **RC** and **RStab** respectively. Lastly, for any value $x \in \mathbb{R}$ we use the notation $(x)^+$ to denote that we are taking the maximum of x and zero.

2.1. Master Problem

The model we employed for the decomposition was the multi-commodity flow formulation that we presented in Bagger et al. (2016). We begin by describing this model, and then we describe how we reformulated it using Benders' decomposition.

For each course $c \in \mathcal{C}$ and period $p \in \mathcal{P}$ there is a binary variable $x_{c,p}$ which is set to one, if c is scheduled in p, and zero otherwise. To calculate the violation of the requested minimum working days we need to calculate which days the courses are scheduled for. A binary variable $t_{c,d}$ is defined for each course $c \in \mathcal{C}$ and day $d \in \mathcal{D}$ which is set to one, if c is scheduled in at least one of the periods $\mathcal{P}_d$, and zero otherwise. We let w_c for each course $c \in \mathcal{C}$ be an integer variable, counting the number of days, below the minimum requested, that c has lectures scheduled. Lastly for each curriculum $q \in \mathcal{Q}$ and period $p \in \mathcal{P}$ there is a binary variable $s_{q,p}$ that is set to one if q has an isolated lecture in p, and zero otherwise. We can then formulate the time scheduling part of the problem as follows:

$$\sum_{p \in \mathcal{P}} x_{c,p} = L_c \quad \forall c \in \mathcal{C} \tag{6}$$

$$x_{c,p} \leq F_{c,p} \quad \forall c \in \mathcal{C}, p \in \mathcal{P} \tag{7}$$

$$\sum_{c \in \mathcal{C}_\gamma} x_{c,p} \leq 1 \quad \forall \gamma \in \Gamma, p \in \mathcal{P} \tag{8}$$

$$\sum_{p \in \mathcal{P}_d} x_{c,p} \geq t_{c,d} \quad \forall c \in \mathcal{C}, d \in \mathcal{D} \tag{9}$$

$$\sum_{d \in \mathcal{D}} t_{c,d} + w_c \geq M_c \quad \forall c \in \mathcal{C} \tag{10}$$

$$\sum_{c \in \mathcal{C}_q} \left(x_{c,p} - \sum_{p' \in \theta_p} x_{c,p'} \right) \leq s_{q,p} \quad \forall q \in \mathcal{Q}, p \in \mathcal{P} \tag{11}$$

$$x_{c,p} \in \mathbb{B} \quad \forall c \in \mathcal{C}, p \in \mathcal{P} \tag{12}$$

$$t_{c,d} \in \mathbb{B} \quad \forall c \in \mathcal{C}, d \in \mathcal{D} \tag{13}$$

$$w_c \in \mathbb{Z}^+ \quad \forall c \in \mathcal{C} \tag{14}$$

$$s_{q,p} \in \mathbb{B} \quad \forall q \in \mathcal{Q}, p \in \mathcal{P} \tag{15}$$

Constraints (6) and (7) ensure that all lectures are scheduled, that they are scheduled in different periods, and that they are only scheduled in the periods where the courses are available. Constraints (8) ensure that at most one lecture from each course-clique is scheduled in every period. The violation of the **MWD** constraint is calculated in (9) and (10). Lastly, the constraints (11) calculate in which periods the curricula have isolated lectures. The domains of the variables are given in (12) – (15).

For the room assignment portion of the problem we define an integer variable $y_{c,r}$ for each course $c \in \mathcal{C}$ and room $r \in \mathcal{R}$ which counts the number of times that c has been assigned to r. $z_{c,r}$ is a binary variable that is set to one if the course $c \in \mathcal{C}$ is assigned to room $r \in \mathcal{R}$ at least once, and zero otherwise. The room assignment portion can then be formulated as follows:

$$\sum_{r \in \mathcal{R}} y_{c,r} = L_c \quad \forall c \in \mathcal{C} \tag{16}$$

$$y_{c,r} \leq L_c z_{c,r} \quad \forall c \in \mathcal{C}, r \in \mathcal{R} \tag{17}$$

$$y_{c,r} \geq z_{c,r} \quad \forall c \in \mathcal{C}, r \in \mathcal{R} \tag{18}$$

$$\sum_{r \in \mathcal{R}} z_{c,r} \geq 1 \quad \forall c \in \mathcal{C}, r \in \mathcal{R} \tag{19}$$

$$y_{c,r} \in \mathbb{Z}^+ \quad \forall c \in \mathcal{C}, r \in \mathcal{R} \tag{20}$$

$$z_{c,r} \in \mathbb{B} \quad \forall c \in \mathcal{C}, r \in \mathcal{R} \tag{21}$$

Constraints (16) ensure that all lectures are assigned a room. For every course $c \in \mathcal{C}$ and room $r \in \mathcal{R}$ the variable $y_{c,r}$ has to be set to zero if $z_{c,r}$ is zero and it has to be set to a positive value if $z_{c,r}$ is set to one. This is ensured by the constraints (17) and (18). Since every lecture has to be scheduled, every course has to be assigned to at least one room, which is reflected by the constraints (19). The

domains of the variables are given in (20) – (21).

Finally, from the two parts; (6) – (15) and (16) – (21), we can formulate the objective function which is to be minimized:

$$\begin{aligned} o(x,y,z) := & W^{\mathbf{MWD}} \sum_{c\in\mathcal{C}} w_c \\ & + W^{\mathbf{RC}} \sum_{c\in\mathcal{C},r\in\mathcal{R}} (S_c - C_r)^+ y_{c,r} \\ & + W^{\mathbf{RStab}} \sum_{c\in\mathcal{C}} \left(\sum_{r\in\mathcal{R}} z_{c,r} - 1 \right) \\ & + W^{\mathbf{IL}} \sum_{q\in\mathcal{Q},p\in\mathcal{P}} s_{q,p} \end{aligned} \tag{22}$$

Until now, we treated the time scheduling and room assignment parts of the problem as separate entities. Given a binary variable $f_{c,p,r}$ for each course $c \in \mathcal{C}$, period $p \in \mathcal{P}$ and room $r \in \mathcal{R}$, which is set to one, if c is scheduled in room r at period p, the two parts can be connected as follows:

$$\sum_{p\in\mathcal{P}} f_{c,p,r} = y_{c,r} \quad \forall c \in \mathcal{C}, r \in \mathcal{R} \tag{23}$$

$$\sum_{r\in\mathcal{R}} f_{c,p,r} \leq x_{c,p} \quad \forall c \in \mathcal{C}, p \in \mathcal{P} \tag{24}$$

$$\sum_{c\in\mathcal{C}} f_{c,p,r} \leq 1 \quad \forall r \in \mathcal{R}, p \in \mathcal{P} \tag{25}$$

$$f_{c,p,r} \in \mathbb{B} \quad \forall c \in \mathcal{C}, p \in \mathcal{P}, r \in \mathcal{R} \tag{26}$$

We showed in Bagger et al. (2016) that the integrality requirements (26) can be relaxed to non-negativity constraints such that the f variables are continuous. This means that we can project them out and replace (23) – (26) by the following Benders' cuts:

$$\sum_{c\in\mathcal{C},p\in\mathcal{P}} \alpha^k_{c,p} x_{c,p} + \sum_{c\in\mathcal{C},r\in\mathcal{R}} \beta^k_{c,r} y_{c,r} \leq \eta^k \quad \forall k \in \mathcal{K} \tag{27}$$

The set $\mathcal{K}$ is the set of Benders' cuts. For each cut $k \in \mathcal{K}$ the coefficient of the variable $x_{c,p}$ is $\alpha^k_{c,p}$ for each course $c \in \mathcal{C}$ and $p \in \mathcal{P}$, and for each course $c \in \mathcal{C}$ and room $r \in \mathcal{R}$ the coefficient of the variable $y_{c,r}$ is $\beta^k_{c,r}$. The parameter η^k is a constant for each $k \in \mathcal{K}$. Since $\mathcal{K}$ can be exponentially large we added the cuts dynamically when required. We added some of the cuts statically. For instance, we cannot schedule more courses to a single period than the number of rooms available:

$$\sum_{c\in\mathcal{C}} x_{c,p} \leq |\mathcal{R}| \quad \forall p \in \mathcal{P} \tag{28}$$

Similarly the total number of times that some room is assigned to by the courses cannot exceed the total number of periods:

$$\sum_{c\in\mathcal{C}} y_{c,r} \leq |\mathcal{P}| \quad \forall r \in \mathcal{R} \tag{29}$$

The cuts (28) and (29) were added to the master problem before we began the solution process, and we generated the remaining cuts when an integer solution violated them.

2.2. Sub-problems

In this section, we describe the sub-problems. Our first sub-problem was based on the problem connecting the x and y variables (23) – (26). These cuts are necessary and sufficient to ensure that the solutions obtained in our decomposed model are equivalent to the solutions in our non-decomposed model. We based the second sub-problem on the indirect connection between the x and z variables. The applicability of the second sub-problem was to generate cuts on the z variables as they are connected with the y in the constraints (17) and (18) so the cuts also had an indirect effect on the y variables.

2.2.1. Dual LP

The first sub-problem we used, directly follows from the LP relaxation of (23) – (26), as previously mentioned, the integrality constraints can be neglected. Given a solution $(\overline{x}, \overline{y}, \overline{z})$ we fix the x and y variables and if (23) – (26) is infeasible then we need to identify a cut. It can be verified that due to constraints (6) and (16) if we remove constraints (25) then the problem is feasible. Hence, to learn if (23) – (26) is infeasible for the given solution, we subtract a non-negative variable u from the left-hand-side of each of the constraints (25). We then search for the solution that minimizes u:

$$\min \quad u \tag{30}$$

$$\text{s.t.} \quad \sum_{p\in\mathcal{P}} f_{c,p,r} = y_{c,r} \quad \forall c \in \mathcal{C}, r \in \mathcal{R} \tag{31}$$

$$\sum_{r\in\mathcal{R}} f_{c,p,r} = x_{c,p} \quad \forall c \in \mathcal{C}, p \in \mathcal{P} \tag{32}$$

$$\sum_{c\in\mathcal{C}} f_{c,p,r} - u \leq 1 \quad \forall r \in \mathcal{R}, p \in \mathcal{P} \tag{33}$$

$$f_{c,p,r} \geq 0 \quad \forall c \in \mathcal{C}, p \in \mathcal{P}, r \in \mathcal{R} \tag{34}$$

$$u \geq 0 \tag{35}$$

Note that we changed the inequality (24) to the equality (32). This is because any solution $(\overline{x}, \overline{y}, \overline{z}, \overline{f})$ that is feasible has to meet equality in the constraints (24), since $L_c = \sum_{r\in\mathcal{R}} \overline{y}_{c,r} = \sum_{p\in\mathcal{P},r\in\mathcal{R}} \overline{f}_{c,p,r} \leq \sum_{p\in\mathcal{P}} \overline{x}_{c,p} = L_c$ must hold for each $c \in \mathcal{C}$. The problem (23) – (26) is feasible for the given solution $(\overline{x}, \overline{y}, \overline{z})$ if, and only if, there exists a solution for (30) – (35) where the variables x and y are fixed at the values $\overline{x}$ and $\overline{y}$ respectively and where

the value of u is zero. We can, therefore, use this problem to generate feasibility cuts.

Given a solution $(\overline{x}, \overline{y}, \overline{z})$, we fix the variables x and y in (30) – (35) to the values $\overline{x}$ and $\overline{y}$, i.e., setting the bounds $\overline{x} \leq x \leq \overline{x}$ and $\overline{y} \leq y \leq \overline{y}$, and solve the problem to optimality. Let $\overline{r}^x_{c,p}$ and $\overline{r}^y_{c,r}$ be the reduced costs of the variables x and y respectively and let $\overline{u}$ be the value of the variable u. According to Fischetti (2015) we can derive the Benders' cut as follows:

$$\sum_{c \in \mathcal{C}, p \in \mathcal{P}} \overline{r}^x_{c,p} \left(x_{c,p} - \overline{x}_{c,p}\right) + \sum_{c \in \mathcal{C}, r \in \mathcal{R}} \overline{r}^y_{c,r} \left(y_{c,r} - \overline{y}_{c,r}\right) + \overline{u} \leq 0 \tag{36}$$

We added the cut (36) whenever it was violated by an integer solution $(\overline{x}, \overline{y}, \overline{z})$.

2.2.2. Maximum Flow / Minimum Cut

In the previous section we described how to obtain the cuts from (23) – (26). Those cuts connect only the x and y variables together. In this section, we look at the link between the x and z variables. Though they are not directly connected in any constraints, they have an indirect relation since the y and z variables are contained in some of the same constraints. Consider the model (23) – (26). It can be shown that the constraints (24) must be met by equality in any feasible solution. When changing the inequality in (24) into an equality, then the equality in (23) can be changed to a less-than-or-equal inequality. Combining this with constraint (17) we have that:

$$\sum_{p \in \mathcal{P}} f_{c,p,r} \leq L_c z_{c,r} \quad \forall c \in \mathcal{C}, r \in \mathcal{R} \tag{37}$$

The consequence of (37) is that for some course $c \in \mathcal{C}$ and some room $r \in \mathcal{R}$, if $z_{c,r}$ is zero, then $f_{c,p,r}$ has to be zero for each period $p \in \mathcal{P}$. This leads to the following model:

$$f_{c,p,r} \leq z_{c,r} \quad \forall c \in \mathcal{C}, p \in \mathcal{P}, r \in \mathcal{R} \tag{38}$$

$$\sum_{r \in \mathcal{R}} f_{c,p,r} = x_{c,p} \quad \forall c \in \mathcal{C}, p \in \mathcal{P} \tag{39}$$

$$\sum_{c \in \mathcal{C}} f_{c,p,r} \leq 1 \quad \forall r \in \mathcal{R}, p \in \mathcal{P} \tag{40}$$

$$f_{c,p,r} \geq 0 \quad \forall c \in \mathcal{C}, p \in \mathcal{P}, r \in \mathcal{R} \tag{41}$$

Note that model (38) – (41) is decomposable by the periods, i.e., to learn if the model is feasible we can consider one period at a time. We can formulate this problem as a maximum flow problem. Given the solution $(\overline{x}, \overline{y}, \overline{z})$ construct a graph $\overline{\mathcal{G}}_p = \left(\overline{\mathcal{V}}_p, \overline{\mathcal{A}}_p\right)$ for each period $p \in \mathcal{P}$. The graph contains a source node (u) and a sink node (v). Furthermore there is a node (c) for each course $c \in \mathcal{C}$ and a node (r) for each room $r \in \mathcal{R}$. For each course $c \in \mathcal{C}$ there is an arc $(u, c) \in \overline{\mathcal{A}}_p$ where the capacity is set to $\overline{x}_{c,p}$. For each room $r \in \mathcal{R}$ there is an arc $(r, v) \in \overline{\mathcal{A}}_p$ with a capacity of one. For each course $c \in \mathcal{C}$ and room $r \in \mathcal{R}$ there is an arc $(c, r) \in \overline{\mathcal{A}}_p$ where the capacity is set to $\overline{z}_{c,r}$. An example of the graph, for an instance with two courses and two rooms for some period $p \in \mathcal{P}$, is illustrated in Figure 2.

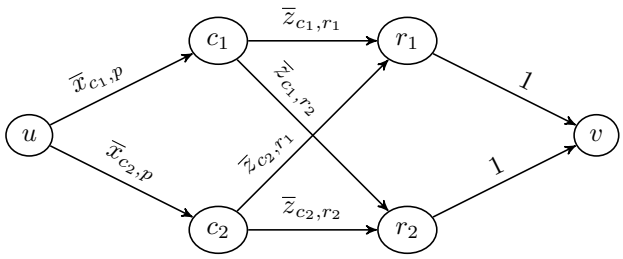

Figure 2: Illustration of the maximum flow graph $\overline{\mathcal{G}}_p = (\overline{\mathcal{V}}_p, \overline{\mathcal{A}}_p)$ for period $p \in \mathcal{P}$ of an instance with two courses and two rooms to validate the $(\overline{x}, \overline{z})$ pair. The labels above the arcs are the corresponding capacities.

As we state in the next proposition 1, we can use this graph to check whether or not (38) – (41) is feasible for a given solution.

Proposition 1. *The model (38) – (41) is feasible for a given solution $(\overline{x}, \overline{y}, \overline{z})$ if, and only if, for each period $p \in \mathcal{P}$ there exists some flow in the graph $\overline{\mathcal{G}}_p$ where the value is equal to $\sum_{c \in \mathcal{C}} \overline{x}_{c,p}$*

Proof of proposition 1. Consider the solution $(\overline{x}, \overline{y}, \overline{z})$ and the graph $\overline{\mathcal{G}}_p$ for some period $p \in \mathcal{P}$. We first prove that if the model is feasible for the period, then there exists some flow in the graph where the value of the flow is equal to $\sum_{c \in \mathcal{C}} \overline{x}_{c,p}$. Subsequently, we prove that if there exists some flow in the graph where the value of the flow is equal to $\sum_{c \in \mathcal{C}} \overline{x}_{c,p}$ then there is a feasible solution for the model for the period.

Consider some solution $\overline{f}$ to the model (38) – (41) for $(\overline{x}, \overline{y}, \overline{z})$. Consider the graph $\overline{\mathcal{G}}_p$ for some period $p \in \mathcal{P}$. For each course $c \in \mathcal{C}$ and room $r \in \mathcal{R}$ the amount of flow we send on the path $(u) \to (c) \to (r) \to (v)$ is equal to $\overline{f}_{c,p,r}$. To validate that this is a flow for the graph we need to check if the node balancing constraints and the capacity constraints are met. Since we are only sending flow on paths from the source to the sink, then the node balancing constraints have to be met. The flow leaving node (c) is equal to $\sum_{r \in \mathcal{R}} \overline{f}_{c,p,r}$ and due to the node balancing constraints and the constraints (39) then the flow on the arc (u, c) must be equal $\overline{x}_{c,p}$ which is the capacity of that arc. The flow on the arc (c, r) is $\overline{f}_{c,p,r}$ and due to constraint (38) this is less than or equal to $\overline{z}_{c,r}$, which is the capacity on the arc. The total amount of flow entering the node (r) is equal to $\sum_{c \in \mathcal{C}} \overline{f}_{c,p,r}$ and due to the node balancing constraint, the flow on the arc (r, v) has to be equal to this value. Due to the constraint (40), the flow on the arc (r, v) has to, therefore, be less than or equal to one which is the capacity of the arc. The value of the flow has to be equal to $\sum_{c \in \mathcal{C}} \overline{x}_{c,p}$ because of the constraints (39). Hence, if (38) – (41) is feasible for the given solution then

there is a flow for the graph where the value of the flow is $\sum_{c\in\mathcal{C}} \overline{x}_{c,p}$.

Now consider some flow for the graph $\overline{\mathcal{G}}_p$ for period $p \in \mathcal{P}$ where the value of the flow is equal to $\sum_{c\in\mathcal{C}} \overline{x}_{c,p}$. For each course $c \in \mathcal{C}$ and room $r \in \mathcal{R}$ let $\overline{f}_{c,p,r}$ denote the amount of flow on the arc (c, r). $\overline{f}$ is a feasible solution for the f variables if the constraints (38) – (40) are not violated. Since the capacity on the arc (c, r) for course $c \in \mathcal{C}$ and room $r \in \mathcal{R}$ is $\overline{z}_{c,r}$ then the constraints (38) cannot be violated. Since the sum of the capacities leaving the source node is equal to $\sum_{c\in\mathcal{C}} \overline{x}_{c,p}$, this means that the flow on arc (u, c) for each course $c \in \mathcal{C}$ has to be equal to $\overline{x}_{c,p}$ and due to the node balancing constraint, all the flow leaving (c) has to also equal this value, i.e., $\sum_{r\in\mathcal{R}} \overline{f}_{c,p,r}$ has to be equal to $\overline{x}_{c,p}$, meaning that the constraints (39) are fulfilled. The total amount of flow entering node (r) has to be equal to $\sum_{c\in\mathcal{C}} \overline{f}_{c,p,r}$, and due to the node balancing constraint, this value cannot exceed one, which means that the constraints (40) are not violated. Hence, if there exists some flow for the graph where the value of the flow is equal to $\sum_{c\in\mathcal{C}} \overline{x}_{c,p}$ then the model is feasible.

Since we considered an arbitrary period, this concludes our proof of proposition 1. □

For the source node (u) and sink node (v) a $u-v$ cut is a cut that separates the graph into two parts; the u side, which contains the source node, and the v side that contains the sink node. For the maximum flow problem, the value of any flow in the graph is upper bounded by the sum of the capacities for any $u-v$ cut (Ahuja et al., 2013, property 6.1). In the remainder of the section, whenever we mention a cut we refer to a $u-v$ cut unless specified otherwise. If we consider some cut $\overline{\chi}_p \subseteq \overline{\mathcal{A}}_p$ in the graph $\overline{\mathcal{G}}_p$ then the sum of the capacities in the cut has to be greater than or equal to $\sum_{c\in\mathcal{C}} \overline{x}_{c,p}$ for the solution to be feasible. Hence, given the solution $(\overline{x}, \overline{y}, \overline{z})$ we solved the maximum flow problem by the labeling algorithm described by Ahuja et al. (2013, chapter 6). This algorithm also finds a minimum cut $\overline{\chi}_p \subseteq \overline{\mathcal{A}}_p$ and if the value of the maximum flow (capacity of the minimum cut) was less than $\sum_{c\in\mathcal{C}} \overline{x}_{c,p}$, we added the following cut to the master problem:

$$\sum_{(u,c)\in\overline{\chi}_p} x_{c,p} + \sum_{(c,r)\in\overline{\chi}_p} z_{c,r} + \left|(r,v) \in \overline{\chi}_p\right| \geq \sum_{(u,c)\in\overline{\mathcal{A}}_p} x_{c,p} \tag{42}$$

For ease of notation in (42) we write $(u, c) \in \mathcal{A}$ which means; for each $c \in \mathcal{C}$ where the arc (u, c) is in the set $\mathcal{A}$. Similarly, $(r, v) \in \mathcal{A}$ means for each room $r \in \mathcal{R}$ where the arc (r, v) is in the set $\mathcal{A}$ and $(c, r) \in \mathcal{A}$ is the set of all the arcs (c, r) for each $c \in \mathcal{C}$ and $r \in \mathcal{R}$ in the set $\mathcal{A}$. We use the notation $|\cdot|$ on these sets to denote the cardinalities, and use this notation throughout the remainder of the paper.

We can rewrite (42) into a simpler form:

$$\sum_{(u,c)\in\overline{\mathcal{A}}_p\setminus\{\overline{\chi}_p\}} x_{c,p} - \sum_{(c,r)\in\overline{\chi}_p} z_{c,r} \leq \left|(r,v) \in \overline{\chi}_p\right| \tag{43}$$

Whenever we obtained an integer solution $(\overline{x}, \overline{y}, \overline{z})$, we added the cut (43) for each period $p \in \mathcal{P}$ where it was violated.

3. Primal Heuristic

The solutions obtained were often infeasible during the Branch-and-Bound process, which meant that it was difficult for the MIP solver to achieve the upper bounds used for pruning. Therefore, we implemented a heuristic to regain feasibility whenever a cut was generated, to assist the solver. In the heuristic we exploited the property that the courses can be assigned to any room. Due to constraints (28) we know that given any solution $(\overline{x}, \overline{y}, \overline{z})$ the time-schedule part $\overline{x}$ of the solution is feasible, meaning that it is possible to create a room assignment based on the time-schedule to obtain a complete solution. The heuristic runs through every period and in each period the lectures that are scheduled in that period are each assigned to a distinct room. This will always be possible since no more courses are scheduled in any period than the number of rooms available. Based on this assignment, the heuristic calculates the values of the y and z variables.

For each period $p \in \mathcal{P}$ we found a minimum cut $\overline{\chi}_p$ in the graph $\overline{\mathcal{G}}_p$ as described in section 2.2.2. Then we calculated the value $v_p\left(\overline{\chi}_p\right)$:

$$v_p\left(\overline{\chi}_p\right) := \sum_{(u,c)\in\overline{\mathcal{A}}_p\setminus\{\overline{\chi}_p\}} \overline{x}_{c,p} - \sum_{(c,r)\in\overline{\chi}_p} \overline{z}_{c,r} - \left|(r,v) \in \overline{\chi}_p\right| \tag{44}$$

If $v_p\left(\overline{\chi}_p\right) > 0$ then we cannot schedule all lectures in p, and we need to change the values of the z variables to obtain feasibility. We did this by ordering the periods according to the values of $v_p\left(\overline{\chi}_p\right)$ from the largest to the smallest. We then iterated over the periods in the given order and for each period $p \in \mathcal{P}$ we performed the following steps:

1. Let the graph $\overline{\mathcal{G}}_p$ be defined as in section 2.2.2 with the same capacities. For each course $c \in \mathcal{C}$ and room $r \in \mathcal{R}$ let the weight on the arc $(c, r) \in \overline{\mathcal{A}}_p$ be $W^{\mathbf{RC}}\left(S_c - C_r\right)^+$ and let the weights of the remaining arcs be zero.
2. If $v_p\left(\overline{\chi}_p\right) \leq 0$, then stop.
3. For every arc $(c, r) \in \overline{\chi}_p$, if $\overline{z}_{c,r}$ (the capacity) is zero then we *open the room*, i.e., we change the value to one.
4. We then find the minimum cost maximum flow $\overline{f}$ (minimum cut $\overline{\chi}_p$) and for every arc (c, r) were we

changed the capacity in step 3, if there is no flow on the arc, i.e., if $\overline{f}_{c,p,r} = 0$, we change the capacity back to zero.

5. Given the new minimum cut $\overline{\chi}_p$, update the value $v_p\left(\overline{\chi}_p\right)$ and go back to step 2.

When we had processed all periods, we set the values $\overline{y}$ as follows:

$$\overline{y}_{c,r} = \sum_{p \in \mathcal{P}} \overline{f}_{c,p,r} \quad c \in \mathcal{C}, p \in \mathcal{P}, r \in \mathcal{R} \tag{45}$$

We then set the $\overline{z}$ values as follows to ensure that the constraints (18) were fulfilled:

$$\overline{z}_{c,r} = \begin{cases} 1 & \text{if } \overline{y}_{c,r} > 0 \\ 0 & \text{otherwise} \end{cases} \quad c \in \mathcal{C}, p \in \mathcal{P}, r \in \mathcal{R} \tag{46}$$

In step 3 a minimum cost maximum flow problem is solved, which is to find a minimum-cost flow among all the maximum flows. The reason we solved a minimum cost maximum flow problem, rather than just a maximum flow problem was that we wished to create a feasible solution without increasing the penalty of the **RStab** constraint too much. The algorithm we used is a modified version of the algorithm described by Ahuja et al. (2013, section 9.7) which finds the minimum cost flow given the demands and supplies on the nodes. We had a supply on the source node, and a demand on the sink node both equal to $\sum_{c \in \mathcal{C}} \overline{x}_p$ and on the rest of the nodes, the demands and supplies were zero. The modification to the algorithm is the stopping criterion. In its pure form, the algorithm assumes that it is possible to meet all the demands of the nodes. Since we did not always have a feasible flow, we added the stopping criterion that if no path exists from the source node to the sink node in the residual graph, then the algorithm has to stop. Hence, the modified algorithm finds the minimum cost flow among all the maximum flows that exists. We found the minimum cut from this flow in the same way as we did in section 2.2.2.

We need to show that the algorithm is finite and returns a feasible solution. If the algorithm assigns all lectures to rooms in each period, then the overall solution is feasible. Therefore, we only need to show that the algorithm is finite and returns a feasible solution for a single period since there is a finite number of periods. For period $p \in \mathcal{P}$, if all the arcs $(c, r) \in \overline{\mathcal{A}}_p$ have a capacity of one, then all lectures scheduled in the period can be assigned to a room. Hence, if the period is infeasible, then some of the arcs $(c, r) \in \overline{\mathcal{A}}_p$ have to have a capacity of zero. As long as at least one of the lectures cannot be assigned to a room, i.e., as long as $v_p\left(\overline{\chi}_p\right) > 0$, the algorithm changes at least one value $\overline{z}_{c,r}$ from zero to one, for some arc (c, r) in each iteration, assuming step 3 is correct. The correctness of step 3 is given by proposition 2. Since there is a finite number of these arcs, then the algorithm also has to be finite. Furthermore, since the algorithm does not stop as long as $v_p\left(\overline{\chi}_p\right) > 0$, then the solution will also be feasible for the period $p \in \mathcal{P}$, since all capacities on the arcs $(c, r) \in \overline{\mathcal{A}}_p$ are set to one in the worst case.

Proposition 2. *For a solution $(\overline{x}, \overline{y}, \overline{z})$, a period $p \in \mathcal{P}$ and a minimum cut $\overline{\chi}_p$, if $v_p\left(\overline{\chi}_p\right) > 0$ then there has to exist a course $c \in \mathcal{C}$ and a room $r \in \mathcal{R}$ where the arc (c, r) is in the cut $\overline{\chi}_p$ and the capacity is zero, i.e., $\overline{z}_{c,r} = 0$*

To prove proposition 2 we first prove that in the minimum cut $\overline{\chi}_p$, at least one of the arcs $(c, r) \in \overline{\mathcal{A}}_p$ has to be in the cut. Subsequently, we prove that the arcs $(c, r) \in \overline{\chi}_p$ cannot all have a capacity of one.

Proof. At least one arc $(c, r) \in \overline{\mathcal{A}}_p$ is in the cut $\overline{\chi}_p$. We begin the proof by showing that not all of the arcs $(u, c) \in \overline{\mathcal{A}}_p$ can be in the minimum cut $\overline{\chi}_p$, nor can all the arcs $(r, v) \in \overline{A}_p$. In Figure 3 we give an illustration of the graph $\mathcal{G}_p$ and two cuts; $\overline{\chi}_p^1$ and $\overline{\chi}_p^2$.

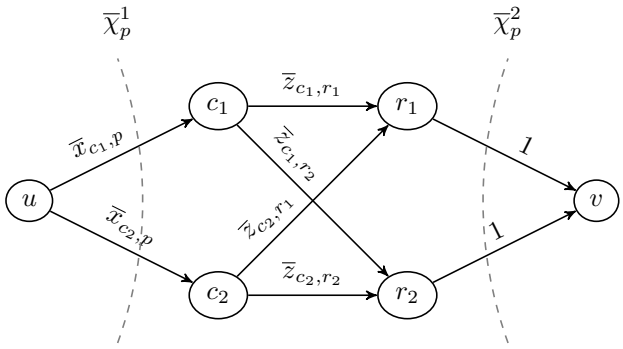

Figure 3: Illustration of the two cuts; cut $\overline{\chi}_p^1$ containing all arcs leaving the source node and $\overline{\chi}_p^2$ containing all arcs entering the sink node.

The cut $\overline{\chi}_p^1$ contains all the arcs $(u, c) \in \overline{\mathcal{A}}_p$ and the cut $\overline{\chi}_p^2$ contains all the arcs $(r, v) \in \overline{\mathcal{A}}_p$. Since we know $v_p\left(\overline{\chi}_p\right) < 0$ for the minimum cut $\overline{\chi}_p$ then neither $\overline{\chi}_p^1$ nor $\overline{\chi}_p^2$ can be a minimum cut. The reason that $\overline{\chi}_p^1$ cannot be a minimum cut is that the total capacity of the minimum cut $\overline{\chi}_p$ is strictly less than $\sum_{c \in \mathcal{C}} \overline{x}_{c,p}$, however, since the cut $\overline{\chi}_p^1$ contains all the arcs $(u, c) \in \overline{\mathcal{A}}_p$ then the total capacity of $\overline{\chi}_p^1$ has to be greater than or equal to $\sum_{c \in \mathcal{C}} \overline{x}_{c,p}$. The reason that $\overline{\chi}_p^2$ cannot be a minimum cut is that $\sum_{c \in \mathcal{C}} \overline{x}_{c,p} \leq |\mathcal{R}|$ due to constraint (28), thus the total capacity of the minimum cut $\overline{\chi}_p$ is strictly less than $|\mathcal{R}|$ since not all of the lectures are assigned to a room. However, since the cut $\overline{\chi}_p^2$ contains all of the arcs $(r, v) \in \overline{\mathcal{A}}_p$ then the total capacity of $\overline{\chi}_p^2$ has to be greater than, or equal to, $|\mathcal{R}|$.

As we cannot have all arcs $(u, c) \in \mathcal{A}_p$ in the minimum cut, there have to exist some course $c \in \mathcal{C}$ where the node (c) is on the same side of the cut as the source node (u). Similarly, as we cannot have all arcs $(r, v) \in \mathcal{A}_p$ in the minimum cut, there has to exist some room $r \in \mathcal{R}$ where the node (r) is on the same side of the cut as the sink

node (v). Hence, the arc (c, r) must be in the cut which concludes our proof. □

We showed that at least one of the arcs $(c, r) \in \overline{\mathcal{A}}_p$ has to be in the minimum cut. One side effect of our proof is that since not all of the arcs $(u, c) \in \overline{\mathcal{A}}_p$ can be in the cut, we have the following:

$$\left|(u, c) \in \overline{\chi}_p\right| \leq |\mathcal{C}| - 1 \tag{47}$$

Similarly, since not all of the arcs $(r, v) \in \overline{\mathcal{A}}_p$ can be in the minimum cut, we have the following:

$$\left|(r, v) \in \overline{\chi}_p\right| \leq |\mathcal{R}| - 1 \tag{48}$$

Before the next proof, we make the assumption that $|\mathcal{R}| \geq 2$. It is fair to make this assumption since if $|\mathcal{R}| = 0$ the problem is infeasible and if $|\mathcal{R}| = 1$ then at most one course is scheduled in each period and can be assigned to the single room.

Proof. $\overline{z}_{c,r} = 0$ *for at least one arc* $(c, r) \in \overline{\chi}_p$. We showed in the previous proof that there is at least one arc (c, r) in the minimum cut for some course $c \in \mathcal{C}$ and room $r \in \mathcal{R}$. Now we show for at least one of the arcs $(c, r) \in \overline{\chi}_p$ that the capacity is zero. To prove this, we assume that every arc $(c, r) \in \overline{\chi}_p$ has a capacity of one, and show that this leads to a contradiction.

Since we consider an infeasible solution, due to constraint (43), the following have to hold:

$$\sum_{(c,r)\in\overline{\chi}_p} \overline{z}_{c,r} + \left|(r, v) \in \overline{\chi}_p\right| < \sum_{(u,c)\in\overline{\mathcal{A}}_p\setminus\overline{\chi}_p} \overline{x}_{c,p} \tag{49}$$

All capacities in the graph are either zero or one; hence, we can calculate an upper bound of the right-hand side of (49) as follows:

$$\sum_{(u,c)\in\overline{\mathcal{A}}_p\setminus\overline{\chi}_p} \overline{x}_{c,p} \leq \left|(u, c) \in \overline{\mathcal{A}}_p\setminus\overline{\chi}_p\right| = |\mathcal{C}| - \left|(u, c) \in \overline{\chi}_p\right| \tag{50}$$

If we insert (50) into (49) together with our assumption that the capacities of all the arcs $(c, r) \in \overline{\chi}_p$ are one then we obtain the following:

$$\left|(c, r) \in \overline{\chi}_p\right| + \left|(r, v) \in \overline{\chi}_p\right| < |\mathcal{C}| - \left|(u, c) \in \overline{\chi}_p\right| \tag{51}$$

Consider an arc (c, r) for some course $c \in \mathcal{C}$ and some room $r \in \mathcal{R}$. For this arc to be in the cut, the node (c) has to be on the same side of the cut as the source node, and the node (r) has to be on the same side as the sink node. The node (c) is on the source side if, and only if, the arc (u, c) is not in the cut. Similarly, the node (r) is on the sink side if and only if the arc (r, v) is not in the cut. So we can express $\left|(c, r) \in \overline{\chi}_p\right|$ as the product of the arcs (u, c) and (r, v) that are not in the cut:

$$\left|(c, r) \in \overline{\chi}_p\right| = \left|(u, c) \in \overline{\mathcal{A}}_p\setminus\overline{\chi}_p\right| \left|(r, v) \in \overline{\mathcal{A}}_p\setminus\overline{\chi}_p\right| \tag{52}$$

We can rewrite (52) by noting that the number of arcs leaving the source, which are not in the cut is equal to the number of arcs leaving the source minus the arcs in the cut, and similarly for the arcs entering the sink:

$$\left|(c, r) \in \overline{\chi}_p\right| = \left(|\mathcal{C}| - \left|(u, c) \in \overline{\chi}_p\right|\right) \left(|\mathcal{R}| - \left|(r, v) \in \overline{\chi}_p\right|\right) \tag{53}$$

Expanding the expression gives us the following:

$$\begin{aligned}\left|(c, r) \in \overline{\chi}_p\right| =& |\mathcal{C}||\mathcal{R}| + \left|(u, c) \in \overline{\chi}_p\right| \left|(r, v) \in \overline{\chi}_p\right| \\ &- |\mathcal{R}| \left|(u, c) \in \overline{\chi}_p\right| - |\mathcal{C}| \left|(r, v) \in \overline{\chi}_p\right|\end{aligned} \tag{54}$$

Substituting (54) into (51) gives the following:

$$\begin{aligned}|\mathcal{C}||\mathcal{R}| + \left|(r, v) \in \overline{\chi}_p\right| < |\mathcal{C}| - \left|(u, c) \in \overline{\chi}_p\right| \left|(r, v) \in \overline{\chi}_p\right| \\ + |\mathcal{R}| \left|(u, c) \in \overline{\chi}_p\right| + |\mathcal{C}| \left|(r, v) \in \overline{\chi}_p\right| - \left|(u, c) \in \overline{\chi}_p\right|\end{aligned} \tag{55}$$

Next we rearrange the terms:

$$\begin{aligned}|\mathcal{C}| \left(|\mathcal{R}| - 1\right) < \left(|\mathcal{C}| - 1\right) \left(|\mathcal{R}| - 1\right) \\ - \left(\left(|\mathcal{R}| - 1\right) - \left|(r, v) \in \overline{\chi}_p\right|\right) \left(\left(|\mathcal{C}| - 1\right) - \left|(u, c) \in \overline{\chi}_p\right|\right)\end{aligned} \tag{56}$$

An upper bound can be found for the right-hand side of (56) by finding a lower bound for the expression:

$$\left(\left(|\mathcal{R}| - 1\right) - \left|(r, v) \in \overline{\chi}_p\right|\right) \left(\left(|\mathcal{C}| - 1\right) - \left|(u, c) \in \overline{\chi}_p\right|\right) \tag{57}$$

Since we know that $\left|(r, v) \in \overline{\chi}_p\right| \leq |\mathcal{R}| - 1$ then $\left(\left(|\mathcal{R}| - 1\right) - \left|(r, v) \in \overline{\chi}_p\right|\right) \geq 0$ has to hold. Similarly $\left(\left(|\mathcal{C}| - 1\right) - \left|(u, c) \in \overline{\chi}_p\right|\right) \geq 0$ also has to hold, which means that the minimum value that (57) can take has to be zero. Hence, the right-hand side of (56) has to be less than or equal to $\left(|\mathcal{C}| - 1\right) \left(|\mathcal{R}| - 1\right)$, and since we assumed that $|\mathcal{R}| \geq 2$ we can divide by $|\mathcal{R}| - 1$ to obtain the following:

$$|\mathcal{C}| < |\mathcal{C}| - 1 \tag{58}$$

As (58) is a contradiction then we have concluded our proof and at least one of the arcs $(c, r) \in \overline{\chi}_p$ must have a capacity of zero. □

We have now proven that our algorithm is finite and returns a feasible solution. We ran this heuristic whenever

we retrieved an integer solution where a cut was generated.

4. Computational Experiments

In this section, we report our computational experiments. We conducted all our tests on a 3.40GHz Intel® Core™ processor running Windows 10 with 8GB memory RAM. We used Gurobi 6.5.2 provided by Gurobi Optimization, Inc. (2016) both for the master problem and also for the subproblem described in section 2.2.1. We set all parameters in Gurobi to their default except, *Presolve*, *Threads* and *LazyConstraints*. We set Presolve to 2 (the most aggressive level), *Threads* to 1 as this was the maximum allowed threads for the ITC2007 competition and *LazyConstraints* to 1 to be able to add cuts on integer solutions.

We tested our approach on four datasets; TEST, COMP, DDS and ERLANGEN. The TEST dataset consists of four data instances, test1 – test4, proposed by Di Gaspero and Schaerf (2003). The COMP dataset contains 21 data instances, comp01 – comp21, described in Di Gaspero et al. (2007) mainly taken from the University of Udine. The DDS dataset contains seven data instances, DDS1 – DDS7, taken from other Italian universities. The ERLANGEN dataset consists of six instances, provided by Dr.-Ing. Moritz Mühlenthaler. All the datasets can be retrieved from `http://tabu.diegm.uniud.it/ctt/index.php`. For the ITC2007 competition, a benchmarking tool was provided, which can be obtained from `http://www.cs.qub.ac.uk/itc2007`. The tool calculates the time limit for the competition on the machine the tool is executed on. This time limit is referred to as one CPU unit, which in our case is 208 seconds. We have written all our implementations (including the flow algorithms) in C#.

The flow diagram in Figure 4 is an illustration of our implementation of Benders' Decomposition.

At first the model is initialized as (6) – (22), (28), (29) and we set the best known solution value to be infinity. We then start Gurobi's Branch-and-Bound solver and in a callback function we retrieve every integer solution that is obtained. Whenever we receive an integer solution we generate the cut (36) and cuts (43) for each period if the solution violates them. If we generate at least one cut, then we run the heuristic described in section 3 to repair the solution. If the repaired solution is better than our current best-known solution, then we reinsert the solution into Gurobi. The reason that we reinsert the solution is that Gurobi can use the solution to prune nodes in the Branch-and-Bound tree and the heuristics can use the solution to search for improvements.

In the following we compare our implementation with results obtained in the literature by other MIP-based methods: Lach and Lübbecke (2012), Burke et al. (2010), Hao and Benlic (2011), and Cacchiani et al. (2013). In section 4.1 we first compare the lower bounds obtained by our implementation with Lach and Lübbecke (2012), Burke et al. (2010), Hao and Benlic (2011), and Cacchiani et al. (2013) on the first fourteen data instances of COMP. Afterwards, we compare the lower bounds obtained on all 21 data instances in COMP as well as all instances from the sets DDS and TEST with Cacchiani et al. (2013) as they have reported lower bounds for all these sets. In section 4.2 we compare both lower and upper bounds obtained by Lach and Lübbecke (2012) and Burke et al. (2010) as these methods provide not only lower bounds but also upper bounds. In section 4.3 we compare our implementation to the original model without Benders' decomposition, i.e., model (6) – (26) where the integrality requirements (26) are relaxed such that the f variables are non-negative continuous variables.

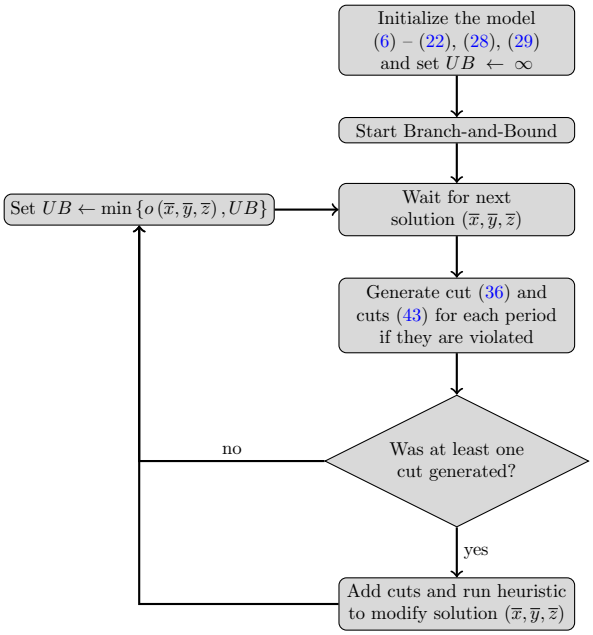

Figure 4: Flow diagram of the implemention of Benders' decomposition. The algorithm continues until a stopping criterion is met, e.g., when a time limit is reached or the best solution is proven optimal.

Before these comparisons we compare the implementation of Benders' decomposition with the results from Bagger et al. (2016) when given a time limit of forty CPU units in Table 1; the minimum cost flow formulation (Min) and the multi-commodity flow formulation (Mult). If an approach obtains a lower bound for an instance which is at least as good as the other approaches for the same instance, then we denote this using a bold font. If the lower bound is better than all the other approaches, then we mark it with an underline. In the bottom of the tables, we report the number of times that each approach obtains a lower bound which is at least as good as the others (the line *Best*) and the number of times a lower bound which is better than all the other approaches (the line following *Best*). Furthermore, we compute the ranking of the approaches on each instance. For each instance, the approach that obtains the best bound is given rank 1, the second best is given rank 2 and so on. If multiple ap-

proaches are tied then they receive the average rank of the respective ranks, e.g., if two approaches are tied for rank 3 and 4 then they both receive rank 3.5 as this is the average. In the last line of the tables, we report the average of the ranks of each approach. The approach that obtains the best rank is marked with a bold font. We use this notation throughout all the tables.

In Table 1 we see that the implementation of Benders' decomposition obtains a lower bound which is at least as good as the other two in 27 instances and a bound which is better in 11 out of 32 instances. Referring to Bonutti et al. (2016) we observe that for two of the instances (marked with an * in the table), comp12 and test4, the lower bounds we obtain, 147 and 49, is an improvement of the currently best-known bounds of 142 and 46 respectively. The upper bounds that we obtain in Benders' decomposition is never better than the Min nor the Mult approaches and has an upper bound which is equal to at least one of the other on six out of 32 instances. Thus, the current focus of the heuristic on feasibility is not enough to improve the upper bounds. Future work could include the possibility of including improvement techniques after the heuristic has regained feasibility.

4.1. Lower Bounds

In this section we first compare the lower bounds obtained by our approach with Lach and Lübbecke (2012), Burke et al. (2010), Hao and Benlic (2011) and Cacchiani et al. (2013). Following the literature we compare the results using three time limits; one CPU unit, ten CPU units and forty CPU units. We start by comparing our results with Cacchiani et al. (2013) as they are the only ones to report lower bounds for the entire COMP dataset as well as for DDS and TEST for forty CPU units. In Table 2 we compare their results with our approach where we use the same notation as in Table 1.

In Table 2 we see that our approach obtains a bound which is at least as good as the bound obtained by Cacchiani et al. (2013) on 23 of the 32 instances and a better bound on eight of the 32 instances.

Next, we compare the lower bounds obtained by our approach with Lach and Lübbecke (2012), Burke et al. (2010), Hao and Benlic (2011), and Cacchiani et al. (2013) on the first fourteen data instances of COMP. The reason for only comparing the first fourteen instances is that these were the only ones that were available for Lach and Lübbecke (2012) and Burke et al. (2010). In the tables 3, 4 and 5 we report the lower bounds obtained for each approach given a time limit of one, ten and forty CPU units respectively. We use the same notation as in Table 1.

In Table 3 – 5 we see that our approach obtains a lower bound which is at least as good as the other approaches in ten instances for one and ten CPU units and in eleven out of the fourteen instances for forty CPU units. Benders' decomposition obtain a better bound in four, three and three out of the fourteen instances for one, ten and forty

Table 1: Comparison of the lower bounds obtained on COMP, DDS and TEST for Min (minimum cost flow formulation from Bagger et al. (2016)), Mult (multi-commodity flow formulation from Bagger et al. (2016)) and Benders (our implementation in this article) with a time limit of forty CPU units. A number in bold font denotes that the approach obtained a value which is at least as good as the other methods for the particular instance. An underlined number means that the approach obtained a value which is better than all the other methods. The line *Best* counts the number of times that each approach obtains a value which is at least as good as the other methods in this table. The line following the line *Best* counts the number of times that the approaches obtained values that are better than all the other methods in this table. The last line *Rank* reports the average rank obtained by each method when comparing with the other methods in this table and the best rank is marked with bold font.

Instance	Min		Mult		Benders	
	LB	UB	LB	UB	LB	UB
comp01	**5**	**5**	**5**	**5**	**5**	**5**
comp02	8	**45**	8	59	**9**	112
comp03	38	123	37	**99**	**42**	107
comp04	**35**	**35**	**35**	**35**	**35**	76
comp05	**186**	**355**	181	377	168	466
comp06	16	**92**	16	**92**	**20**	139
comp07	0	**179**	**6**	-	**6**	183
comp08	**37**	**37**	**37**	41	**37**	99
comp09	74	105	73	**103**	**82**	113
comp10	**4**	**18**	**4**	68	**4**	72
comp11	**0**	**0**	**0**	**0**	**0**	**0**
comp12	142	**423**	140	500	**147***	464
comp13	56	66	**59**	**59**	57	152
comp14	44	**55**	43	74	**48**	93
comp15	38	123	37	**99**	**42**	107
comp16	13	61	11	**42**	**16**	76
comp17	43	123	44	**109**	**46**	132
comp18	**36**	**78**	30	108	35	88
comp19	**56**	**57**	55	**57**	54	92
comp20	**0**	**50**	**0**	96	**0**	340
comp21	56	156	57	**133**	**62**	138
DDS1	**46**	**70**	44	76	45	214
DDS2	**0**	**0**	**0**	**0**	**0**	**0**
DDS3	**0**	**0**	**0**	**0**	**0**	**0**
DDS4	**15**	**17**	**15**	12079	**15**	1192
DDS5	**0**	**0**	**0**	**0**	**0**	**0**
DDS6	**0**	**2**	**0**	27	**0**	123
DDS7	**0**	**0**	**0**	**0**	**0**	**0**
test1	**224**	**224**	**224**	**224**	**224**	314
test2	**16**	**19**	**16**	**19**	**16**	104
test3	**59**	**75**	**59**	**75**	**59**	96
test4	44	**91**	43	107	**49***	101
Best	**19**	**25**	**17**	**19**	**27**	**6**
	4	**13**	**1**	**7**	**11**	
Rank	2.03	**1.59**	2.27	1.84	**1.70**	2.56

Table 2: Comparison of the lower bounds obtained on COMP, DDS and TEST for CCRT13 (Cacchiani et al., 2013) and Benders (our implementation) with a time limit of forty CPU units. The notation follows that of Table 1.

Instance	CCRT13	Benders
comp01	**5**	**5**
comp02	**16**	9
comp03	**52**	42
comp04	**35**	**35**
comp05	166	**168**
comp06	11	**20**
comp07	**6**	**6**
comp08	**37**	**37**
comp09	**92**	82
comp10	2	**4**
comp11	**0**	**0**
comp12	100	**147**
comp13	**57**	**57**
comp14	**48**	**48**
comp15	**52**	42
comp16	13	**16**
comp17	**48**	46
comp18	**52**	35
comp19	48	**54**
comp20	**4**	0
comp21	**68**	62
DDS1	40	**45**
DDS2	**0**	**0**
DDS3	**0**	**0**
DDS4	**17**	15
DDS5	**0**	**0**
DDS6	**0**	**0**
DDS7	**0**	**0**
test1	**224**	**224**
test2	**16**	**16**
test3	**59**	**59**
test4	46	**49**
Best	**24**	**23**
	9	**8**

CPU units respectively. Considering the average rank obtained we see that our approach obtains a lower average rank than all the other approaches on the first fourteen data instances of COMP for all three time limits.

4.2. Upper Bounding Methods

In this section, we compare both the upper bounds that we obtain on the first fourteen data instances from COMP with Lach and Lübbecke (2012) and Burke et al. (2010) as these are the only methods besides ours that obtain both lower and upper bounds. We report the results in the tables 6 – 8 where we use the same notation as in Table 1.

Table 3: Comparison of the lower bounds obtained on the first fourteen data instances of COMP for the different approaches with a time limit of one CPU unit; LL12 (Lach and Lübbecke, 2012), BMPR10 (Burke et al., 2010), HB11 (Hao and Benlic, 2011), CCRT13 (Cacchiani et al., 2013) and Benders (our implementation). The notation follows that of Table 1.

Instance	LL12	BMPR10	HB11	CCRT13	Benders
comp01	4	0	4	**5**	**5**
comp02	0	0	**10**	0	7
comp03	0	25	26	24	**37**
comp04	22	**35**	**35**	**35**	**35**
comp05	92	119	19	6	**145**
comp06	7	13	12	0	**17**
comp07	0	**6**	5	0	**6**
comp08	30	**37**	**37**	**37**	**37**
comp09	37	68	39	**92**	78
comp10	2	3	**4**	0	**4**
comp11	**0**	**0**	**0**	**0**	**0**
comp12	29	**101**	43	0	91
comp13	33	52	46	**57**	56
comp14	40	41	41	32	**43**
Best	**1**	**5**	**5**	**6**	**10**
		1	**1**	**2**	**4**
Rank	4.21	2.71	2.82	3.50	**1.75**

Table 4: Comparison of the lower bounds obtained on the first fourteen data instances of COMP for the different approaches with a time limit of ten CPU units; LL12 (Lach and Lübbecke, 2012), BMPR10 (Burke et al., 2010), HB11 (Hao and Benlic, 2011), CCRT13 (Cacchiani et al., 2013) and Benders (our implementation). The notation follows that of Table 1.

Instance	LL12	BMPR10	HB11	CCRT13	Benders
comp01	4	4	4	**5**	**5**
comp02	8	0	12	**16**	8
comp03	0	33	34	**52**	41
comp04	28	**35**	**35**	**35**	**35**
comp05	25	111	69	6	**167**
comp06	10	15	12	11	**19**
comp07	2	**6**	**6**	**6**	**6**
comp08	34	**37**	**37**	**37**	**37**
comp09	41	65	67	**92**	81
comp10	**4**	**4**	**4**	2	**4**
comp11	**0**	**0**	**0**	**0**	**0**
comp12	32	95	78	0	**144**
comp13	39	52	53	**57**	**57**
comp14	41	42	43	**48**	47
Best	**2**	**5**	**5**	**10**	**10**
				4	**3**
Rank	4.36	3.14	2.86	2.61	**2.04**

Table 5: Comparison of the lower bounds obtained on the first fourteen data instances of COMP for the different approaches with a time limit of forty CPU units; LL12 (Lach and Lübbecke, 2012), BMPR10 (Burke et al., 2010), HB11 (Hao and Benlic, 2011), CCRT13 (Cacchiani et al., 2013) and Benders (our implementation). The notation follows that of Table 1.

Instance	LL12	BMPR10	HB11	CCRT13	Benders
comp01	4	**5**	4	**5**	**5**
comp02	11	1	12	**16**	9
comp03	25	33	36	**52**	42
comp04	28	**35**	**35**	**35**	**35**
comp05	108	114	80	166	**168**
comp06	10	16	16	11	**20**
comp07	**6**	**6**	**6**	**6**	**6**
comp08	**37**	**37**	**37**	**37**	**37**
comp09	46	66	67	**92**	82
comp10	**4**	**4**	**4**	2	**4**
comp11	**0**	**0**	**0**	**0**	**0**
comp12	53	95	84	100	**147**
comp13	41	54	55	**57**	**57**
comp14	46	42	43	**48**	**48**
Best	**4**	**6**	**5**	**10**	**11**
				3	**3**
Rank	4.00	3.32	3.21	2.32	**2.14**

Table 6: Comparison of the upper bounds (UB) obtained on the first fourteen data instances of COMP for the different approaches with a time limit of one CPU units; LL12 (Lach and Lübbecke, 2012), BMPR10 (Burke et al., 2010) and Benders (our implementation). The notation follows that of Table 1.

Instance	LL12	BMPR10	Benders
comp01	**12**	168	14
comp02	239	**114**	118
comp03	194	**158**	179
comp04	**44**	153	108
comp05	965	1447	**604**
comp06	395	277	**227**
comp07	525	-	**260**
comp08	**78**	173	106
comp09	115	**112**	142
comp10	235	**70**	101
comp11	7	288	**0**
comp12	1122	-	**874**
comp13	**98**	556	229
comp14	113	123	**103**
Best	**4**	**4**	**6**
	4	**4**	**6**
Rank	2.00	2.36	**1.64**

Regarding the upper bounds we see that in Table 6 we obtain upper bounds which are better on six out of the fourteen instances and the two other approaches both obtain a better upper bound on four of the instances each. For both ten (Table 7) and forty (Table 7) CPU units our approach obtains a better upper bound for three of the instances, Lach and Lübbecke (2012) obtains a better up-

Table 7: Comparison of the upper bounds (UB) obtained on the first fourteen data instances of COMP for the different approaches with a time limit of ten CPU units; LL12 (Lach and Lübbecke, 2012), BMPR10 (Burke et al., 2010) and Benders (our implementation). The notation follows that of Table 1.

Instance	LL12	BMPR10	Benders
comp01	12	10	**5**
comp02	**93**	101	112
comp03	**86**	144	122
comp04	41	**36**	76
comp05	**468**	649	581
comp06	**79**	317	139
comp07	**28**	857	191
comp08	**48**	53	99
comp09	**106**	115	113
comp10	**44**	49	72
comp11	7	12	**0**
comp12	657	889	**472**
comp13	**67**	92	222
comp14	**54**	72	93
Best	**10**	**1**	**3**
	10	**1**	**3**
Rank	**1.36**	2.43	2.21

Table 8: Comparison of the upper bounds (UB) obtained on the first fourteen data instances of COMP for the different approaches with a time limit of forty CPU units; LL12 (Lach and Lübbecke, 2012), BMPR10 (Burke et al., 2010) and Benders (our implementation). The notation follows that of Table 1.

Instance	LL12	BMPR10	Benders
comp01	12	9	**5**
comp02	**46**	63	112
comp03	**66**	123	107
comp04	38	**36**	76
comp05	**368**	629	466
comp06	51	**46**	139
comp07	**25**	45	183
comp08	44	**41**	99
comp09	**99**	105	113
comp10	**16**	23	72
comp11	7	12	**0**
comp12	548	785	**464**
comp13	**66**	67	152
comp14	**53**	55	93
Best	**8**	**3**	**3**
	8	**3**	**3**
Rank	**1.50**	2.07	2.43

per bound on ten and 8 instances respectively and Burke et al. (2010) obtains a better upper bound on one and three instances respectively.

4.3. Large Datasets (Erlangen)

In this section, we test our model on the ERLANGEN dataset. The data instances in this set are larger than for the other datasets we have tested and for some of the instances even solving the root note LP takes longer time than the forty CPU units we tested on. Therefore we have tested them using 100 CPU units, and we have changed the parameter setting to use the non-homogeneous barrier method as the LP solver (Method=2, NodeMethod=2 and BarHomogeneous=0) and with the number of threads to be equal to the number of processors (Threads=4) as this accelerates the solution time of the LP. We compare our implementation to the original model (the model that we have based our decomposition on) to see the benefits of the decomposition for such large datasets.

In Table 9 we report the lower and upper bounds obtained after running the MIP solver for the given time limit (100 CPU) for both the original model and the decomposition. In the last two columns we report how much the values are improved by Benders. In the column *LB* we report the increase of the lower bound in percent which is calculated as $(LB_{Benders} - LB_{Mult})/LB_{Mult}$ where LB_{Mult} and $LB_{Benders}$ are the lower bounds obtained by the original and the decomposed model respectively. Similarly, in column *UB* we report the decrease of the upper bound.

Table 9: Comparison of the lower bound (LB) and upper bound (UB) obtained for the original model (Mult) and our implementation (Benders) on the ERLANGEN dataset when running the Branch-and-Bound solver Gurobi. The last two columns (Impr.) report how much the bound is improved by Benders in percentages.

	MULT		Benders		Impr.	
Instance	LB	UB	LB	UB	LB	UB
2011_2	2105	10390	**2385**	**10127**	13%	3%
2012_1	1235	21178	**1393**	**18334**	13%	13%
2012_2	1556	-	**1766**	**21722**	13%	-
2013_1	1259	172474	**1417**	**28529**	13%	83%
2013_2	1086	-	**1274**	**19808**	17%	-
2014_1	834	30985	**975**	**18851**	17%	39%
Avg.					14%	35%

We see in Table 9 that the decomposition approach obtains better bounds for all six instances, where for two of the instances the original model does not obtain any solution. Our decomposition gives an improvement of 14% and 35% on average for the lower and upper bounds respectively. However, it should be noted that the average improvement on the upper bounds is calculated on only four of the six insrances

In Table 10 we report the performance for the root LP, i.e., the results of solving the LP relaxation in the root node before the MIP solver starts to add cuts and branch. For both the original model and the decomposition we report the time spent by the solver in seconds (Time) and the objective value of the LP (Obj). In the last two columns we report how much the values are improved be Benders. In the column, *Time* we report the speed-up factor which we calculate as the time that the solver spent on solving the root node LP in the original model divided by the time spent in the decomposed model. In the column *Obj* we report the increase of the objective value in permille which is calculated as $(Obj_{Benders} - Obj_{Mult})/Obj_{Mult}$ where Obj_{Mult} and $Obj_{Benders}$ are the objective values of the root node LP in the original and the decomposed model respectively.

Table 10: Comparison of the time (Time) it takes to solve the root node LP and the objective value (Obj) obtained for the original model (Mult) and our implementation (Benders) on the ERLANGEN dataset. The last two columns (Impr.) report how much the values are improved by Benders. The improvement of the time is reported as the speed-up factor and the improvement of the objective value is reported in permille. The last line (Avg.) report the average improvements.

	Mult		Benders		Impr.	
Instance	Time	Obj	Time	Obj	Time	Obj
2011_2	5836.7	1985.7	**86.6**	**1994.8**	×57	5‰
2012_1	2518.9	1015.8	**117.7**	**1019.8**	×21	4‰
2012_2	6306.5	1347.9	**162.3**	**1347.9**	×39	0‰
2013_1	3083.1	1030.9	**153.5**	**1032.4**	×20	1‰
2013_2	5634.6	959.9	**199.9**	**967.7**	×28	8‰
2014_1	2740.6	734.9	**224.8**	**737.6**	×12	4‰
Avg.					×31	4‰

In Table 10 we see that our decomposition speeds up the solution time for the root node LP by more than thirty times on average. This speed-up makes sense as the majority of the variables in the model are projected out in the decomposition. What may come as a surprise is that the objective value for five of the instances is larger in the decomposition than in the original model. The reason for this improvement is that the MIP solver manages to heuristically generate some integer solutions for the decomposition before it starts to solve the LP. These integer solutions are infeasible and some of the cuts we generate are the cuts (43) which are based on more information than what is available for the original model, i.e., these cuts are based on model (39) – (41). In our opinion, these results illustrate that Benders' decomposition is a powerful tool for this problem, especially for massive datasets.

5. Conclusion

In this article, we proposed a Benders' decomposition on a Mixed Integer Programming (MIP) model for the Curriculum-Based Course Timetabling problem. We also described a heuristic to repair the solutions that were cut away by the Benders' feasibility cuts. Regarding lower bounds, our approach obtained better results on average compared to other MIP based approaches for the first fourteen instances, from the second international timetabling

competition. We compared our approach on a larger set of data (a total of 32 instances) with the state-of-the-art algorithm. We obtained a lower bound, which was at least good on 23 of the 32 instances and we obtained a better bound on eight instances. Furthermore, we improved the currently best known lower bounds for two out of the 32 instances. Benders' decomposition did not provide better upper bounds than the non-decomposed model, thus, the focus on feasibility in the heuristic was not sufficient to improve the upper bounds. Lastly, we compared Benders' decomposition to the MIP model, where the decomposition originated from, on six additional data instances that are much larger than the other 32 instances. The results revealed the benefit of implementing Benders' decomposition as both the lower and upper bounds were improved for all of the instances compared to the original model. To the best of our knowledge, no other studies have tested MIP based methods on these data instances; hence, we improved the best known lower bounds for all six instances.

Acknowledgments

The authors would like to thank Assistant Professor Evelien van der Hurk for her valuable feedback toward the improvement of the manuscript.

Funding: This work is part of an industrial PhD project funded by Innovation Fund Denmark (IFD). The IFD has supported this work solely financially, and did not take part in any research related activities.

References

Ahuja, R., Magnanti, T., Orlin, J., 2013. Network Flows: Theory, Algorithms, and Applications. Always learning. Pearson Education Limited.
URL https://books.google.ca/books?id=rFuLngEACAAJ

Bagger, N.-C. F., Simon, K., Sørensen, M., Stidsen, T. R., 2016. Flow formulations for curriculum-based course timetabling. Available on Optimization Online.
URL http://www.optimization-online.org/DB_HTML/2016/12/5786.html

Benders, J., 1962. Partitioning procedures for solving mixed-variables programming problems. Numerische Mathematik 4 (1), 238–252.
URL http://dx.doi.org/10.1007/BF01386316

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., April 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, L. D., Schaerf, A., 2016. Curriculum-based course timetabling. http://tabu.diegm.uniud.it/ctt/index.php [Last visited 30/12-2016].

Burke, E. K., Mareček, J., Parkes, A. J., Rudová, H., 2008. Penalising patterns in timetables: Novel integer programming formulations. In: Kalcsics, J., Nickel, S. (Eds.), Operations Research Proceedings 2007. Vol. 2007 of Operations Research Proceedings. Springer Berlin Heidelberg, pp. 409–414, 10.1007/978-3-540-77903-2_63.

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

Burke, E. K., Mareček, J., Parkes, A. J., 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, Queenes University SARC Building, Belfast, United Kingdom.

Di Gaspero, L., Schaerf, A., 2003. Multi-neighbourhood local search with application to course timetabling. In: Burke, E., De Causmaecker, P. (Eds.), Practice and Theory of Automated Timetabling IV. Vol. 2740 of Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 262–275.
URL http://dx.doi.org/10.1007/978-3-540-45157-0_17

Fischetti, M., 2015. The simpler the better: Thinning out mip's by occam's razor. Presentation at CORS/INFORMS 2015 Joint International Meeting in Montréal. Slides retrieved December 16, 2016, from: http://www.dei.unipd.it/~fisch/papers/slides/2015%20CORS%20%5bFischetti%20-%20Occam%20Razor%5d.pdf.

Geoffrion, A. M., 1972. Generalized benders decomposition. Journal of optimization theory and applications 10 (4), 237–260.

Gurobi Optimization, Inc., 2016. Gurobi optimizer reference manual.
URL http://www.gurobi.com

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

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

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

Lübbecke, M. E., 2015. Comments on: An overview of curriculum-based course timetabling. TOP 23 (2), 359–361.

Martin, R. K., 1999. Large scale linear and integer optimization: a unifoed approach. Kluwer Academic Publishers.

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