A Cutting Plane Algorithm for the International Timetabling Competition 2019 Problem

Holm, Dennis Søren; Lysgaard, Jens; Stidsen, Thomas Jacob Riis

Publication date:
2022

Document Version
Publisher's PDF, also known as Version of record

A Cutting Plane Algorithm for the International Timetabling Competition 2019 Problem

Dennis S. Holm · Jens Lysgaard · Thomas J. R. Stidsen

March 2022

Abstract The International Timetabling Competition 2019 (ITC 2019) considers a university course timetabling problem where semester events must be assigned a time and a room while students must be enrolled in classes according to their course registration. Holm et al. (2022) show a graph-based Mixed Integer Programming (MIP) formulation of the ITC 2019 problem. The MIP model uses various graph structures to give a strong formulation. However, the MIP model is still hard to solve for many of the 30 ITC 2019 instances. To generate high-quality solutions, Mikkelsen and Holm (2022) use a parallelized fix-and-optimize matheuristic based on the graph-based MIP model. The matheuristic found the best solutions to 29 of 30 ITC 2019 instances during the competition and was the winning algorithm. Even though the algorithm won the competition superiorly, only five instances are solved to optimality. This technical report aims to research the possibilities of having an exact method for solving the ITC 2019 instances. We do this by investigating the Linear Programming (LP) relaxation of the graph-based MIP model and introducing groups of constraints that might be beneficial to use as cuts. This report will focus on solving the LP relaxation with an iterative cutting plane algorithm. The search for a usable cutting plane algorithm provides two new lower bounds on the ITC 2019 instances.

Keywords Mixed Integer Programming · Cutting plane · University Course Timetabling · International Timetabling Competition 2019 · ITC 2019

Dennis S. Holm's Ph.D. project is part of the Data Science for University Management project (dsumsoftware.com) funded by MaCom A/S and Innovation Fund Denmark. MaCom A/S and Innovation Fund Denmark has supported the work solely financially and has not participated in any research-related activities.

Dennis S. Holm
Technical University of Denmark
DTU Management
Akademivej
Building 358
2800 Kgs. Lyngby
E-mail: dsho@dtu.dk; guldfiskenholm@gmail.com

Jens Lysgaard
Aarhus University
Department of Economics and Business Economics
Fuglesangs Allé 4
Building 2621, 116
8210 Aarhus V
Tel.: +45 87164898
E-mail: lys@econ.au.dk

Thomas J. R. Stidsen
Technical University of Denmark
DTU Management
Akademivej
Building 358
2800 Kgs. Lyngby
Tel.: +45 45254449
E-mail: thst@dtu.dk

1 Introduction

The International Timetabling Competition 2019 (ITC 2019) is the third of the international timetabling competitions focusing on university timetabling. The first competition, ITC2002, used a typical simplified timetabling problem. The second competition, ITC2007, used data from the University of Udine to generate three different tracks: post-enrollment based course timetabling, examination timetabling, and curriculum based course timetabling. The ITC 2019 uses data from 10 different universities on five continents to derive a generalized formulation for the university course timetabling problem. The data is received from the UniTime software, a system that supports the development of educational timetables. Thus, the ITC 2019 data represent real-world data and is the best data available in academia for university timetabling problems.
The ITC 2019 formulation can both support the post-enrollment based and the curriculum based course timetabling problems along with a large variety of other timetable features. Thus, the ITC 2019 presents a problem formulation that might theoretically cover any university timetabling problem of the world.

To solve the university course timetabling problem of the ITC 2019, Mikkelsen and Holm (2022) uses a graph-based Mixed Integer Programming (MIP) model as a basis for a parallelized Fix-and-Optimize (FaO) matheuristic. The algorithm runs several FaO algorithms with different neighborhoods in parallel. The parallel runs share solutions so that no run is "left behind" in a nonpromising area of the solution space. They also introduce a diversification scheme, such that if the search stagnates, the FaO algorithms can start from a new initial solution. Along with the FaO algorithm, they run a MIP model solver (Gurobi) on the full model. This parallelized algorithm shows to be the best solution method to the ITC 2019 since it was the winning algorithm of the competition and, according to the "live score,"[1] still is the leading solution method. Another solution method for the ITC 2019 is presented by Müller (2020) which is one of the organizers of the ITC 2019 and currently second place in the live scoreboard[1]. Müller (2020) uses the UniTime Solver, which is a heuristic approach. The UniTime solver uses an iterative forward search to construct a feasible timetable. The iterative forward search has implemented conflict-based statistics, a scheme of conflicts from previously found solutions that should be avoided in the following solution, thereby preventing cycling. When an initial timetabling is found, the UniTime solver uses a hill climber to improve the timetable until a local optimum is reached. At that point, the Great Deluge (Dueck, 1993) technique is used. The solver has split the student sectioning and timetabling parts such that an assignment of students is used in the class scheduling algorithm, minimizing the number of student conflicts (and other objectives). When the class scheduling algorithm is finished, a local search reassigns students to classes when beneficial.

The teams that placed first and second in the competition used matheuristics (Mikkelsen and Holm, 2022; Rappos et al., 2019), whereas the rest of the top 5 teams used metaheuristic-based approaches (Gashi and Sylejmani, 2019; Er-rhaimini, 2019; Lemos et al., 2019). This paper investigates the possibility of another exact method to solve the ITC 2019 problem. The method in focus is the cutting plane algorithm. Burke et al. (2012) implement a branch-and-cut method for the university course timetabling problem of ITC2007. The branch-and-cut method shows varying performance on the different instances. The benefits include a faster root node solution and improved lower bounds.
We seek to find a relaxed version of the LP relaxation of the graph-based MIP model by Holm et al. (2022), where we add constraints dynamically (cuts), such that we only use a subset of constraints to find a feasible LP solution. If the cutting procedure is found to be faster than solving the full LP model, it should be used for further research into using a branch-and-cut algorithm on this problem.

The paper is organized as follows. Section 2 and 3 describes the ITC 2019 problem and the MIP model, respectively. Section 4 finds the proportion of binding constraints for different constraint types in the LP-relaxed model. Section 5 presents preliminary tests which uses our own lazy-constraint implementation to investigate the solution times for the different constraint types. Section 6 investigates different cutting methods for the student sectioning constraints, which proved most promising in the preliminary tests. Section 7 concludes and presents two new lower bounds.

2 The ITC 2019 Problem

The ITC 2019 problem involves scheduling classes to times and rooms combined with assigning students to classes according to their course registration. The students' class assignments must follow the given course structure. The courses can have different configurations

[1] https://www.itc2019.org/score

where a correctly enrolled student must be assigned one configuration by attending one class of the configuration's subparts. The classes are also presented with a student enrollment limit. Furthermore, some classes might be related in a parent-child manner, where any attendee of the child class must attend the parent class. For the class scheduling part, each class is presented with a set of available rooms (if required) and a set of available times, each with an associated penalty. The rooms may have given distances to other rooms, making it possible to model conflicts where an attendee cannot get to events on the same day as the travel time between the rooms is longer than the break between the events.

The introduction of travel times means that we distinguish between two overlap types. *Time overlap*; when the classes are scheduled on the same day in overlapping times. *Time-room* overlaps; when the classes are scheduled on the same day in rooms with travel time longer than the time between the classes. When we mention *overlap*, we consider both. A *student conflict* is then defined as a student attending two classes that overlap and is penalized by one unit. To model different features of the time table *distribution constraints* are used. The distribution constraints can be either hard or soft and there exists 19 different types, which include `DifferentDay`, `Precedence`, and `SameRoom`. Most distribution constraints consider the classes pairwise, and others consider features of many classes as a whole. The latter are denoted as *special distribution constraints*. The special distribution constraints include `MaxBlock` which limits the length of a block of consecutive classes.

If the university models their problem as a curriculum based course timetabling problem, they use the `SameAttendees` distribution constraint to model the curricula. The `SameAttendees` constraints prevent overlaps (both time and time-room) if it is a hard constraint and penalize the overlaps if it is soft with a specific penalty. The objective function is a combination of the four penalty types, each with an overall weight: room assignment, time assignment, distribution constraint penalty, and students conflicts.

3 The graph-based MIP model

The reader does not need to be introduced in detail to the complete MIP model used in this report, but we introduce required aspects in this section. The MIP model used is the graph-based MIP by Holm et al. (2022). The MIP model uses two main binary decision variables $x_{c,t,r}$ and $e_{s,c}$, one to handle the assignment of classes to times and rooms and one to enroll students into classes.
The primary constraints of the model include that all classes must be assigned, rooms cannot be double-booked, and students must be enrolled correctly into their course registration. The model also includes many auxiliary variables and constraints to control these.
Edge covers of a conflict graph are used to model many distribution constraints. The edge covers use graph structures to cover all edges of the conflict graph at least once. A clique cover is used for the hard constraints as the clique constraints prevent more than one of the variables from being used in a feasible solution. A star cover is used for the soft constraints, as each pair of connected vertices will impose the related penalty to the objective function. Furthermore, the model includes valid inequalities in the form of odd-cycles.
The special distribution constraints are harder to model and are thus handled by separate constraints and auxiliary variables.
The student sectioning part is modeled by constraints that ensure the correct enrollment of students into classes according to course structure, class limits, and parent-child relations, along with constraints to control the student conflict variables.

4 The binding constraints

To find the most promising constraints for the cutting plane algorithm, we search for constraint types, where the proportion of binding constraints is low (and thus the amount of non-binding/unneeded constraints is large). We find the binding constraints by searching for constraints with zero slack variable at the LP optimal solution. To give a better overview of the constraints types, we have grouped them by related types where possible. For example, all the student sectioning constraints are in one group, and all the graph-derived constraints are in another. Table 1 shows the overall proportions of binding constraints for the groups. We have only results for 24 out of 30 instances because the remaining six instances take more than seven days to solve the LP relaxation. A complete overview for each instance is shown in Appendix A. The groups can be described as follows. The *room double book* group contains the hard constraints that prevent room double booking. The *graph constraints* group is composed by the conflict graphs constraints constructed from the distribution constraints; cliques, stars, and odd-cycles. The *SameAttendees* group consider only the time-room overlap of hard and soft `SameAttendees` constraints since the time overlap is included in the conflict graphs. The *SameRoom* group consider the hard and soft `SameRoom` con-

Constraints	Average
Room double booking	4.61%
Graph constraints	12.25%
SameAttendees	13.39%
SameRoom	98.76%
Special	59.99%
Student sectioning	4.68%

Table 1 The average proportion of binding constraints for 24 of the 30 ITC 2019 instances.

straints as they are not part of any graph constraints. *Special* is a group of all the special distribution constraints. *Student sectioning* is the group of student sectioning constraints that are not modeled as equality constraints. This leaves out constraints that ensure the students are assigned according to the course structure, as they are equality constraints.

The grouping gives the advantage that we can process more types together but has a disadvantage when there are large differences within the group. For example, the graph group contains the odd-cycle valid inequality constraints where very few are binding (0.05%). The odd-cycle constraints are grouped with the clique and star cover constraints, where a larger portion is binding (22.86% and 12.84% respectively).

To see the effect of removing the non-binding constraint, we remove the non-binding constraints and solve the resulting relaxed LP models. We compare the solution time of the full LP with the solution times of the versions without the non-binding constraints in Table 2. The full LP models have been solved five times to give an average solution time since this time is what we intend to compare against throughout the test phase. The other LP models have been run just once.

We see from Table 2 that all solved instances containing students will significantly benefit from having only the binding student sectioning constraints in the model. Generally, the having only the binding room double booking constraints is also beneficial. We also see that the graph constraints have some impact on the solution time, especially on the instances without students. This might be because instances without students uses `SameAttendees` constraints for modeling curricula or such. The `SameAttendees` constraints add time overlap edges in the class-time conflict graph. For the instances with special distribution constraints we see a mixed picture, some instances benefit from having only the binding special distribution constraints included, others do not. Removing the non-binding SameRoom constraints will many times result in a slower solution time and held together with the large proportion of binding constraints (Table 1) we exclude the SameRoom constraints from further investigation.

Instance	Full LP	Room double booking	Graph constraints	SameAttendees	SameRoom	Special	Student sectioning
agh-fis-spr17	10,080	3,749	22,865	15,080	13,313	5,109	**792**
agh-ggis-spr17	80	44	60	81	91	114	**35**
*bet-fal17**							
iku-fal17	3,687	**348**	3,831	4,213	4,694	-	-
mary-spr17	442	230	437	363	328	-	**15**
muni-fi-spr16	727	778	223	-	609	540	**7**
muni-fsps-spr17	9	5	6	10	13	-	**1**
*muni-pdf-spr16c**							
pu-llr-spr17	1,083	1,228	346	1,065	1,186	3,305	**21**
tg-fal17	2	1	2	2	3	-	-
agh-ggos-spr17	12,998	**418**	7,004	12,488	71,765	9,667	683
agh-h-spr17	40,074	**7,514**	13,504	62,232	85,480	21,237	30,440
lums-spr18	51	**5**	24	58	51	-	-
muni-fi-spr17	184	2,909	130	-	203	176	**10**
muni-fsps-spr17c	1,929	158	4,080	1,105	3,544	-	**91**
muni-pdf-spr16	535,720	36,814	*time*	873,221	564,384	*time*	**35,558**
nbi-spr18	240	132	191	143	179	-	**32**
pu-d5-spr17	389	239	514	-	369	299	**14**
*pu-proj-fal19**							
yach-fal17	178	37	190	175	170	150	**8**
*agh-fal17**							
*bet-spr18**							
iku-spr18	2,048	58	**1,280**	1,575	1,998	-	-
lums-fal17	857	95	**36**	731	782	-	-
mary-fal18	2,251	905	3,444	3,233	3,138	-	**22**
muni-fi-fal17	284	230	255	-	302	310	**13**
muni-fspsx-fal17	10,754	2,798	8,292	7,630	10,302	-	**329**
*muni-pdfx-fal17**							
pu-d9-fal19	472,555	*time*	*time*	727,675	485,987	**352,037**	*time*
tg-spr18	20	4	**1**	12	12	11	-
Number of solution times faster than the Full LP		20	14	8	7	9	17

Table 2 The solution times (seconds) of the full LP and relaxations of different types, where the non-binding constraints are not present. As an example we see the instance *pu-llr-spr17* takes 1,083 seconds to solve the full LP while only 21 second if only the binding student sectioning constraints are present. The lowest solution time is presented in boldface. One instance *tg-fal17* does not have a clear fastest solution time. The dash (-) means that the instance does not include any constraints of that group. Instances with (*) were not solved within 7 days. *time* indicates that the relaxation took more than 12 days to solve.

5 Preliminary cuts test

We investigate the possibility of a cutting procedure, which in each iteration solves a relaxed LP model and adds all violated constraints of that solution. To see the behavior of the different constraints groups we do preliminary tests. The tests aim is to compare the solution times of the full LP models with the time it takes for a cutting procedure to end. In particular, it is interesting to see how fast the relaxed LP models solve and how many iterations of the cutting procedure can be made and how many are needed. If the relaxed LP models are fast to solve it leaves more time for the cutting. To perform the preliminary cuts tests we will use a simple cutting procedure, where we generate the full LP model with a subset of relaxed constraints. The relaxed constraints will in each iteration be checked for violation and added if violated.

In practice, we relax the constraints by adding or subtracting (depending on if it is a smaller than or larger than constraint) a value P to the right-hand side (RHS) of the constraints, making it impossible to violate the constraint. The constraints considered to be used as cuts are denoted the *cut constraints*. This way, the constraints exist in the LP model but are "inactive." The first iteration solves the LP model where all the cut constraints are inactive. When we have a feasible solution, we reevaluate all the inactive cut constraints with the original RHS. If the constraint is violated, we *activate* it by subtracting/adding P to the RHS. Thereby the constraint is reverted to the original RHS. When all inactive cut constraints have been reevaluated, we re-solve the LP model, warm starting from the previous solution. This iterative process is run until there are no inactive cut constraints to activate, and thus the cutting procedure terminates. The procedure is denoted the *preliminary cutting procedure* (PCP). The focus of PCP is to test the iterative addition of constraints without considering the cut generating algorithm. Since all constraints of the full LP are generated for the PCP, there is no time-saving in the model generation from which a cutting-plane algorithm benefits.
The cut constraints will be the constraint groups from Section 4 where the proportion of binding constraints in the LP solution is relatively low. Thereby, we will consider the groups shown in Table 1 except the SameRoom group.

In this section, we will go through the results of the PCP using each of the groups as cut constraints. We test the PCP on all ITC 2019 instances. We use Gurobi 9.1 as the LP solver. The full LP solution times is an average of five runs with a time limit of 7 days. As the PCP solves more LP models, we set the LP solver time limit to 24 hours for each iteration. If it takes more than 24 hours to solve the relaxed LP model, then it is expected to be too slow to consider. Furthermore, we use a total time limit of 7 days for the whole PCP to finish.
We present each group of cut constraints in a subsection with a recap of what we have discovered when using the PCP with those cut constraints and a table that shows the total solve time, number of iterations, and number of added cuts for each instance. Furthermore, we present a few plots of the iterative movement of the objective values and a marker of the full LP objective and solve time. The plots have been chosen to show a representative view of the runs. We will summarize what we have seen from the results and provide plots of representative instances that show the objective value over time.

Algorithm 1 Preliminary cut procedure (PCP)

procedure PCP()
construct LP
C = list of ($\leq$) constraints to consider
for all constraints c in C **do**
    $c.\text{RHS} = c.\text{RHS} + P$
repeat
    solve relaxed LP
    for all constraints c in C **do**
        if $c.\text{LHS} \geq c.\text{RHS} - P$ **then**
            $c.\text{RHS} = c.\text{RHS} - P$
            remove c from C
until no constraints are violated

5.1 SameAttendees

For the PCP with the SameAttendees group, we see that the PCP compared to the full LP either results in a small improvement in solver time or a rather large increase (Table 3). For two instances the PCP shows a significant decrease in the solver time (*agh-fis-spr17* and *muni-fsps-spr17c*), both reductions by around 1,000 seconds (16.87% and 45.37%, respectively). Some instances takes only one iteration to finish, even though the LP solution showed binding constraints, these instances does not activate any cut constraints. Others take a few iterations but do not change objective value. Figure 1 shows some examples. A few of the instances actually activate constraints at each iteration (where the objective is increased), but they all use more time to finish than the full LP. We believe that the SameAttendees group is not eligible for further investigation by these observations. Thereby all SameAttendees constraints should be active in the LP model.

Instance	Full LP solver time	PCP Solve time	Solve count	Added cuts
agh-fis-spr17	10,079.83	**8,378.38**	2	10
agh-ggis-spr17	**79.86**	85.59	1	0
bet-fal17		time limit		
iku-fal17	**3,686.95**	4,720.73	3	6
mary-spr17	442.44	**398.73**	1	0
muni-fi-spr16	726.68	-	-	-
muni-fsps-spr17	**9.41**	11.18	3	13
muni-pdf-spr16c		time limit		
pu-llr-spr17	**1,083.48**	1,974.13	2	2
tg-fal17	**2.20**	4.11	2	6
agh-ggos-spr17	**12,997.96**	24,238.58	1	0
agh-h-spr17	**40,073.69**	61,247.16	2	3
lums-spr18	51.49	50.31	1	0
muni-fi-spr17	183.93	-	-	-
muni-fsps-spr17c	1,929.30	**1,053.82**	2	2
muni-pdf-spr16	**535,719.89**	time limit		
nbi-spr18	240.05	**220.17**	3	37
pu-d5-spr17	389.12	-	-	-
pu-proj-fal19				
yach-fal17	178.17	178.28	3	7
agh-fal17		time limit		
bet-spr18		time limit		
iku-spr18	**2,048.23**	2,432.75	5	6
lums-fal17	857.03	**736.85**	1	0
mary-fal18	**2,250.80**	2,486.86	2	1
muni-fi-fal17	283.60	-	-	-
muni-fspsx-fal17	**10,753.51**	25,954.44	6	23
muni-pdfx-fal17		time limit		
pu-d9-fal19	472,554.98	time limit		
tg-spr18	20.25	18.97	2	2

Table 3 The time used to solve the full LP model compared to the time used to solve the PCP with SameAttendees cut constraints. The fastest solve time is marked by boldface, when there is a *significant* difference. A dash (-) means that the instance is not relevant for the cutting type as constraints are not present. *Time limit* indicates that the 24 hour Gurobi time limit was reached.

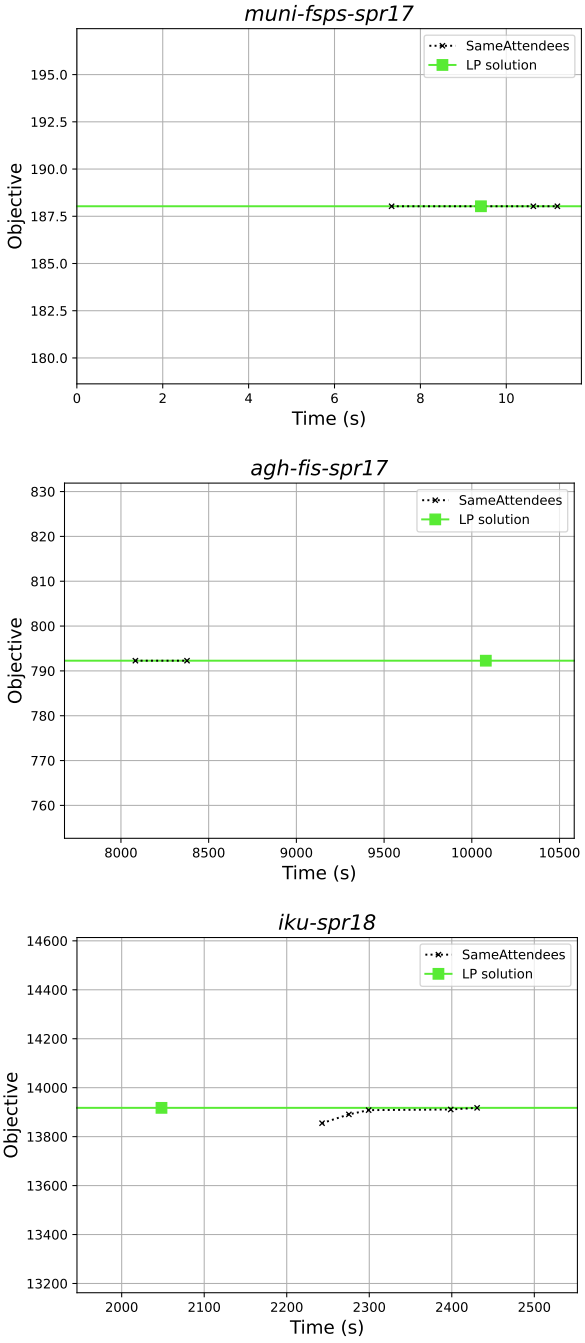

Fig. 1 Three different types of PCP with SameAttendees as cut constraints. For *muni-fsps-spr17* the PCP finds a solution of equal objective to the full LP faster, but has to cut twice and thereby has a longer solution time. The PCP shows to be faster than the full LP for *agh-fis-spr17*. An example with objective change during the PCP is shown by *iku-spr18*.

5.2 Room double booking

When using the PCP with the room double booking cut constraints, we see a similar pattern as the PCP with the SameAttendees cut constraints. Compared to the full LP solution times, we sometimes see a small improvement and other times we see an aggravation, that can be huge. For the PCP double booking cut constraints we see two instances with remarkable improvements; *agh-fis-spr17* and *agh-ggos-spr17* reduced by $4,950.12$ (49.10%) and $9,120.45$ (70.17%), respectively. We also find an instance that is solved, where the full LP model was not solved within seven days. From the plots of the development of objective value over time we see a curve that is slowly converging toward the full LP objective. Figure 2 shows two of the runs. The performance could be related to the room utilization or to a subset of rooms being demanded by many classes, but such conclusion would need more research. The presented results do not provide evidence for further considerations of the room double booking constraints. Thereby all room double booking constraints should be active in the LP model.

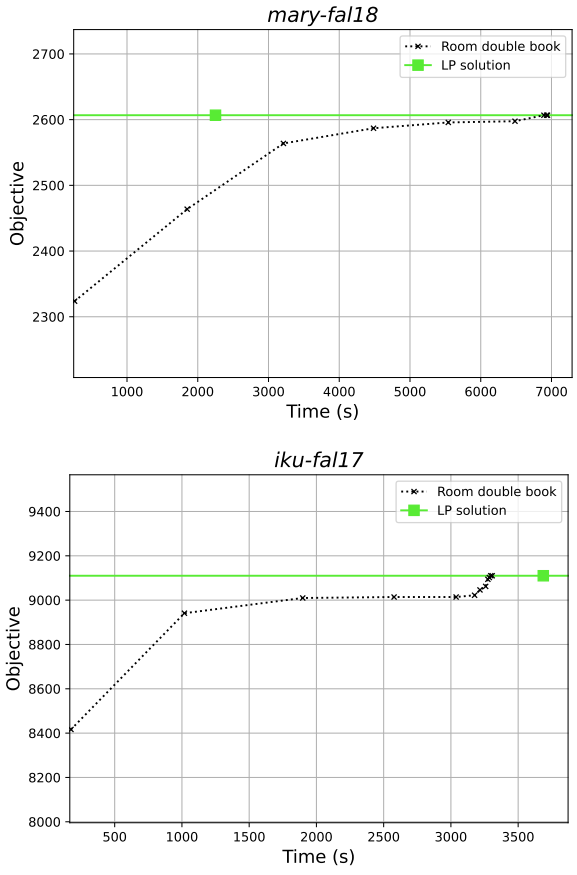

Fig. 2 Some examples of the PCP with room double booking cut constraints.

Instance	Full LP solver time	PCP Solve time	Solve count	Added cuts
agh-fis-spr17	10,079.83	**5,129.71**	8	345
agh-ggis-spr17	**79.86**	260.34	6	620
bet-fal17		time limit		
iku-fal17	3,686.95	**3,309.96**	12	1,796
mary-spr17	**442.44**	978.12	9	619
muni-fi-spr16	**726.68**	35,493.01	10	348
muni-fsps-spr17	**9.41**	42.11	6	219
muni-pdf-spr16c		**140,054.09**	11	647
pu-llr-spr17	**1,083.48**	3,470.17	8	642
tg-fal17	**2.20**	3.39	5	248
agh-ggos-spr17	12,997.96	**3,878.51**	7	389
agh-h-spr17	**40,073.69**	60,752.36	7	327
lums-spr18	**51.49**	114.19	8	419
muni-fi-spr17	**183.93**	1,925.93	8	349
muni-fsps-spr17c	1,929.30	**1,797.08**	6	315
muni-pdf-spr16	**535,719.89**	time limit	2	376
nbi-spr18	240.05	**103.72**	9	585
pu-d5-spr17	**389.12**	37,539.89	12	814
pu-proj-fal19		time limit		
yach-fal17	**178.17**	855.30	5	241
agh-fal17		time limit	2	1,315
bet-spr18		time limit		
iku-spr18	2,048.23	**1,934.28**	8	1,630
lums-fal17	**857.03**	1,016.21	9	380
mary-fal18	**2,250.80**	6,943.97	9	598
muni-fi-fal17	**283.60**	4,827.63	9	379
muni-fspsx-fal17	**10,753.51**	52,279.11	12	896
muni-pdfx-fal17		time limit	2	996
pu-d9-fal19	**472,554.98**	time limit	3	1,260
tg-spr18	20.25	**10.02**	5	278

Table 4 The time used to solve the full LP compared to the time used to solve the PCP with room double booking cut constraints, the number of iterations, and the number of activated cut constraints. The fastest solver time is marked by boldface. *Time limit* indicates that the 24 hour Gurobi time limit was reached.

5.3 Graph constraints

When running the PCP with graph cut constraints, each iteration shows an increase in objective value for almost all instances. Often the first relaxed LP solution is found relatively easily, but the PCP needs many cutting iterations to finish. Most plots show a steadily but slowly converging curve, which finishes long after the full LP model is solved. The plots show large increments in the objective values in the first few iterations, but also that the difference between the first PCP solution and the full LP objective is large. With the full LP model being solved in less time than the PCP takes to finish, it indicates that the full LP model might benefit from having the graph constraints. The conclusion is that it is doubtful that the graph constraint group is suitable to be used as cuts, and thereby the graph constraints should be active in the LP model.

Instance	Full LP solver time	PCP Solve time	Solve count	Added cuts
agh-fis-spr17	**10,079.83**	30,628.32	16	6,047
agh-ggis-spr17	**79.86**	3,680.28	35	19,641
bet-fal17		time limit		
iku-fal17	**3,686.95**	161,990.60	27	7,321
mary-spr17	**442.44**	1,520.88	18	1,468
muni-fi-spr16	**726.68**	2,659.24	13	842
muni-fsps-spr17	**9.41**	52.85	13	1,391
muni-pdf-spr16c		time limit		
pu-llr-spr17	1,083.48	83,412.62	6	326
tg-fal17	**2.20**	8.18	16	4,426
agh-ggos-spr17	**12,997.96**	40,947.76	28	19,544
agh-h-spr17	**40,073.69**	time limit	10	21,759
lums-spr18	**51.49**	151.15	13	1,930
muni-fi-spr17	**183.93**	1,966.65	10	716
muni-fsps-spr17c	**1,929.30**	9,613.75	9	1,906
muni-pdf-spr16	**535,719.89**	time limit		
nbi-spr18	240.05	**227.33**	23	893
pu-d5-spr17	**389.12**	25,530.83	19	2,468
pu-proj-fal19		time limit		
yach-fal17	**178.17**	312.29	8	391
agh-fal17		time limit	2	9,714
bet-spr18		time limit		
iku-spr18	**2,048.23**	5,866.75	16	5,611
lums-fal17	**857.03**	250.40	16	2,893
mary-fal18	**2,250.80**	17,956.20	8	524
muni-fi-fal17	**283.60**	4,407.18	16	1,026
muni-fspsx-fal17	**10,753.51**	time limit	2	1,461
muni-pdfx-fal17		time limit		
pu-d9-fal19	**472,554.98**	time limit		
tg-spr18	20.25	19.38	17	5,051

Table 5 The time used to solve the full LP compared to the time used to solve the PCP with graph cut constraints, the number of iterations, and the number of activated cut constraints. The fastest solver time is marked by boldface. *Time limit* indicates that the 24 hour Gurobi time limit was reached.

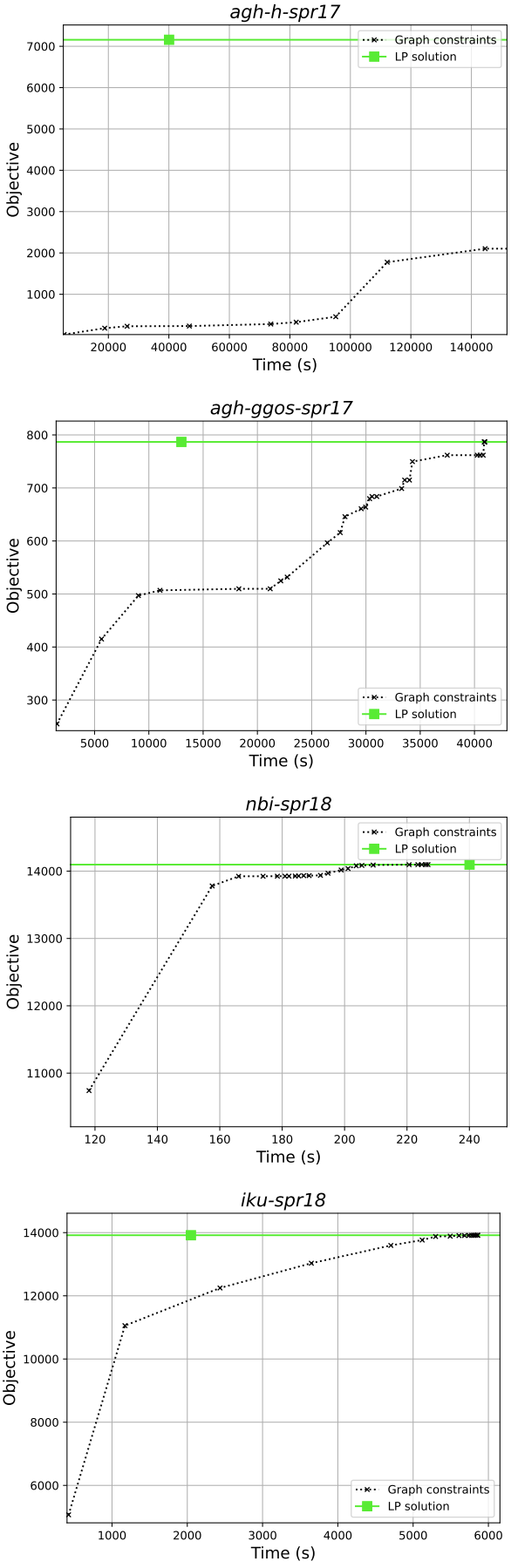

Fig. 3 Some examples of the PCP with graph cut constraints. Many instances starts low on the objective axis and converge towards to the full LP objective with a decreasing slope.

5.4 Special constraints

For these constraints, we must keep in mind that 11 instances do not include any special constraints. For the remaining 19 instances, eight were not solvable within the given time limit of the PCP, which leaves 11 instances solved, where only one (*agh-ggos-spr17*) was significantly faster. However, most of the curves are flat, which means that we are already close to (if not at) the full LP objective by the first iteration. For some instances, it seems that this method looks promising. However, the number and types of binding constraints do not provide any intuitive correlation with the gain from the cutting procedure. Combined with the fact that only 10 instances were solved, we exclude the special constraints for further investigation. The special constraints are thus all active in the LP model.

Instance	Full LP solver time	PCP Solve time	Solve count	Added cuts
agh-fis-spr17	**10,079.83**	14,200.12	10	14,374
agh-ggis-spr17	**79.86**	91.48	4	946
bet-fal17		time limit		
iku-fal17	3,686.95	-	-	-
mary-spr17	442.44	-	-	-
muni-fi-spr16	726.68	**623.23**	5	4,237
muni-fsps-spr17	9.41	-	-	-
muni-pdf-spr16c		time limit		
pu-llr-spr17	**1,083.48**	1,237.29	3	1,050
tg-fal17	2.20	-	-	-
agh-ggos-spr17	12,997.96	**6,443.45**	4	820
agh-h-spr17	**40,073.69**	110,294.81	11	64,826
lums-spr18	51.49	-	-	-
muni-fi-spr17	**183.93**	1,795.07	10	32,421
muni-fsps-spr17c	1,929.30	-	-	-
muni-pdf-spr16	**535,719.89**	time limit		
nbi-spr18	240.05	-	-	-
pu-d5-spr17	**389.12**	2,081.35	6	5,565
pu-proj-fal19		time limit		
yach-fal17	178.17	**131.68**	1	0
agh-fal17		time limit		
bet-spr18		time limit		
iku-spr18	2,048.23	-	-	-
lums-fal17	857.03	-	-	-
mary-fal18	2,250.80	-	-	-
muni-fi-fal17	**283.60**	4,304.87	11	45,935
muni-fspsx-fal17	10,753.51	-	-	-
muni-pdfx-fal17		time limit		
pu-d9-fal19	**472,554.98**	time limit		
tg-spr18	20.25	**12.38**	3	22

Table 6 The time used to solve the full LP compared to the time used by the PCP with special cut constraints, the number of iterations, and the number of activated cut constraints. The fastest solver time is marked by boldface. A dash (-) means that the instance is not relevant for the cutting type as constraints are not present for the instance. *Time limit* indicates that the 24 hour Gurobi time limit was reached.

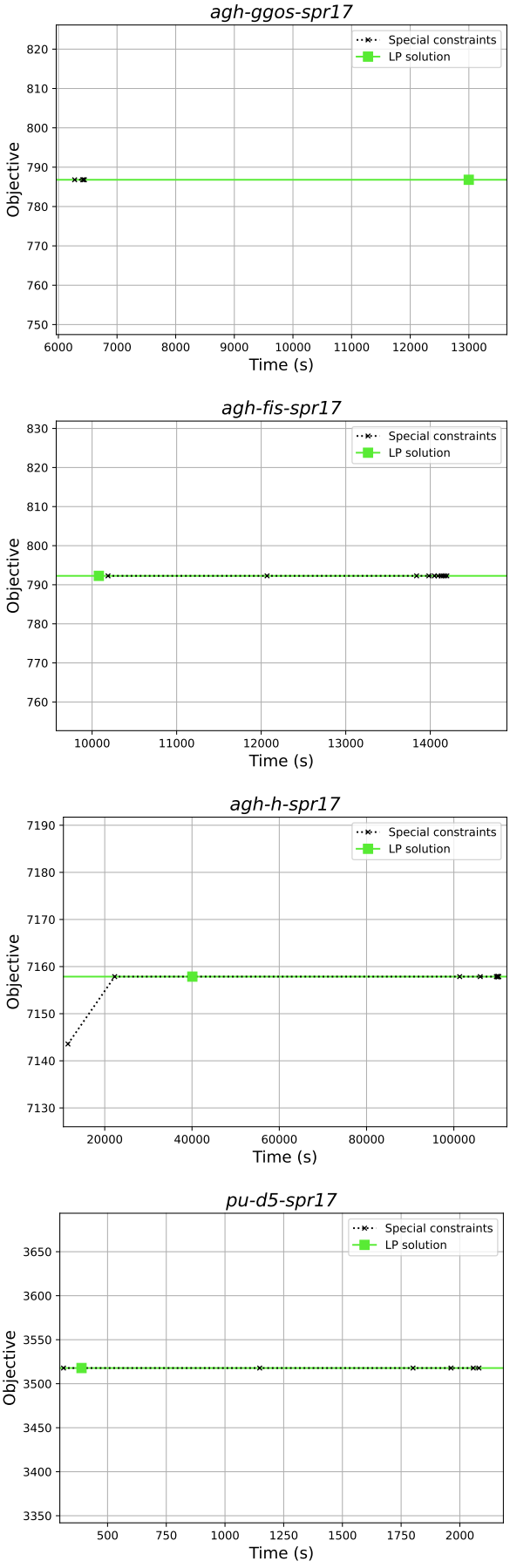

Fig. 4 Examples of four PCP runs with the special cut constraints.

5.5 Student sectioning

The student sectioning constraint group does not affect six of the instances as they do not consider student conflicts. When comparing the full LP and the PCP with student sectioning constraints cuts, we see that some of the solve times are much larger for the PCP. Some instances that are easy for the full LP, *pu-d5-spr17* and *muni-fi-fal17*, are struggling for the PCP. However, we also see the opposite. Two instances not solved by the full LP within seven days, *muni-pdf-spr16c* and *bet-spr18*, are solved in less than 36 hours with the PCP. In general, the PCP with student conflict cut constraints uses a lot of iterations, which comes at the cost of a long computation time. When we look at the plots for the objective values over time, we see that many iterations do not improve the objective. The objective value of the PCP reaches the full LP objective value a long time before the PCP finishes. The long tail of equal objective values can be seen in Figure 5. Because of this, we introduce Table 8 which shows the gap between the PCP objective and the full LP objective at the solution time of the full LP model. Whereas Table 7 only shows that the PCP were faster on two instances compared to the full LP, Table 8 shows that the PCP is at a 0% gap at the time for full LP solution for seven instances. We also see from Table 8 that the average of PCP iterations performed at the time of the full LP solution is 5.5 and produces a gap of 19.7%. The number of iterations to reach a 0% gap and the number of total iterations (Table 7) are on average 9.4 and 150.3, respectively. Because of fast convergence combined with the fact that there are millions of student sectioning constraints, we find this the most promising group of cut constraints.

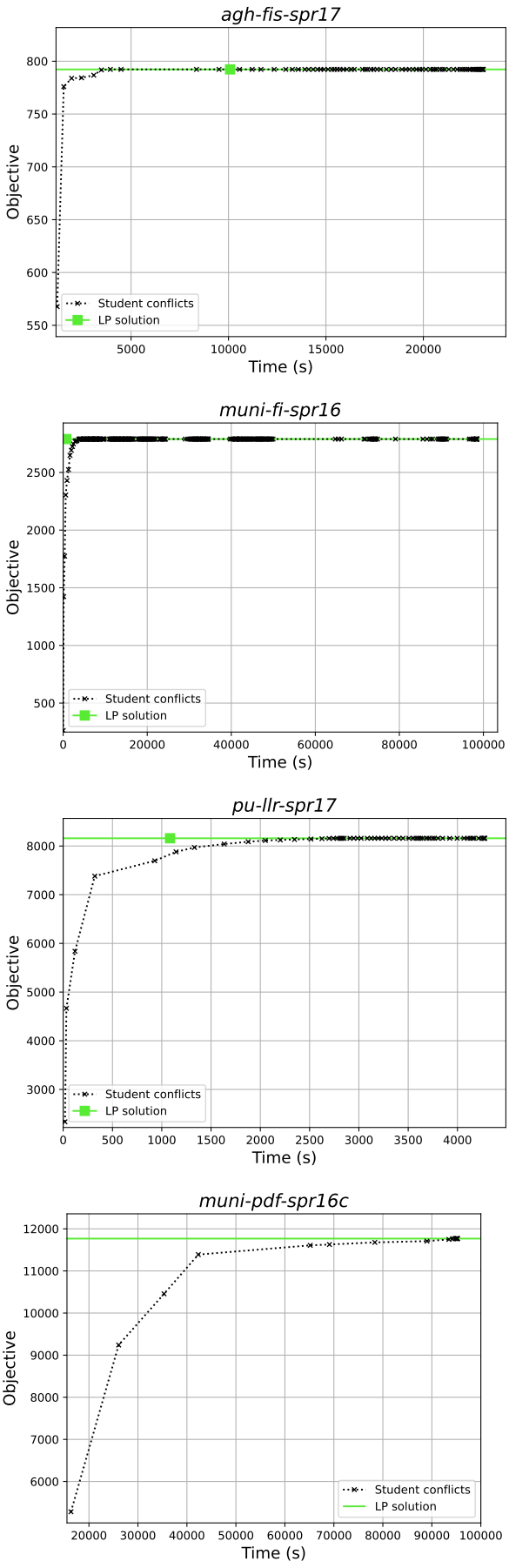

Fig. 5 Examples of the PCP with student sectioning cut constraints. The PCP uses a lot of time on iterations that cuts off solutions with equal objective value for many of the instances.

Instance	Full LP solver time	PCP Solve time	Solve count	Added cuts
agh-fis-spr17	**10,079.83**	23,122.88	79	39,000
agh-ggis-spr17	**79.86**	484.53	15	27,947
bet-fal17		time limit		
iku-fal17	3,686.95	-	-	-
mary-spr17	**442.44**	1,023.38	28	16,979
muni-fi-spr16	**726.68**	98,481.52	391	176,549
muni-fsps-spr17	**9.41**	450.72	117	24,030
muni-pdf-spr16c		**95,255.03**	16	32,490
pu-llr-spr17	**1,083.48**	4,279.90	60	57,540
tg-fal17	2.20	-	-	-
agh-ggos-spr17	**12,997.96**	28,980.19	18	17,855
agh-h-spr17	**40,073.69**	time limit		
lums-spr18	51.49	-	-	-
muni-fi-spr17	**183.93**	52,015.75	406	126,725
muni-fsps-spr17c	**1,929.30**	3,225.93	19	11,465
muni-pdf-spr16	**535,719.89**	time limit	3	17,673
nbi-spr18	**240.05**	358.36	35	10,016
pu-d5-spr17	**389.12**	98,698.11	105	76,411
pu-proj-fal19		time limit		
yach-fal17	**178.17**	2,735.91	60	26,457
agh-fal17		time limit	2	5
bet-spr18		**124,145.95**	21	30,940
iku-spr18	2,048.23	-	-	-
lums-fal17	857.03	-	-	-
mary-fal18	**2,250.80**	12,642.68	47	54,295
muni-fi-fal17	**283.60**	205,223.41	988	216,677
muni-fspsx-fal17	10,753.51	time limit*	6	21,404
muni-pdfx-fal17		time limit		
pu-d9-fal19	**472,554.98**	time limit	2	102,808
tg-spr18	20.25	-	-	-

Table 7 The time used to solve the full LP compared to the time used by the PCP with student conflict cut constraints, the number of iterations, and the number of activated cut constraints. The fastest solver time is marked by boldface. A dash (-) means that the instance is not relevant for the cutting type as constraints are not present for the instance. *Time limit* indicates that the 24 hour Gurobi time limit was reached. *Time limit** indicates that the 7 day total time limit was reached.

	At full LP solution time		
Instance	Gap	Number of iterations	Iterations to 0.0% gap
agh-fis-spr17	0.0%	10	6
agh-ggis-spr17	41.7%	1	6
bet-fal17			
iku-fal17	-	-	-
mary-spr17	0.0%	6	3
muni-fi-spr16	17.4%	4	16
muni-fsps-spr17	0.0%	3	1
muni-pdf-spr16c	0.0%	16	10
pu-llr-spr17	5.7%	5	24
tg-fal17	-	-	
agh-ggos-spr17	2.8%	1	7
agh-h-spr17	100.0%		
lums-spr18	-	-	-
muni-fi-spr17	46.3%	2	14
muni-fsps-spr17c	0.0%	7	1
muni-pdf-spr16	17.7%	2	
nbi-spr18	0.3%	16	30
pu-d5-spr17	2.5%	1	3
pu-proj-fal19			
yach-fal17	0.0%	3	1
agh-fal17			
bet-spr18	0.0%	21	6
iku-spr18	-	-	-
lums-fal17	-	-	-
mary-fal18	12.8%	4	15
muni-fi-fal17	95.2%	1	12
muni-fspsx-fal17	1.4%	1	5
muni-pdfx-fal17			
pu-d9-fal19	50.6%	1	
tg-spr18	-	-	-
Average	19.7%	5.5	9.4

Table 8 Gap and iteration statistics for the PCP with student sectioning cut constraints at the time of full LP solution.

5.6 Conclusion of the preliminary test

The PCP with SameAttendees and room double booking cut constraints show some promising results in finding faster solutions to seven instances. However, the promising results are primarily for instances that are relatively easy to solve for the full LP model. For the rest of the instances, the total number of PCP iterations is low, and the number of iterations performed before the full LP model solution time are also low. Unless there is a way to find finish the PCP in 1-3 iteration(s), these methods are unsuitable.

The graph cut constraints show to be very slow and very far from the solution times of the full LP. For all instances but one, it is better just to solve the full LP model. Therefore, we do not consider these for further investigation.

For the special constraints, we see that 11 instances do not even contain special constraints, and therefore the cutting procedure would be equal to solving the full LP model. For the remaining 19 instances, eight cannot solve the first PCP iteration within 24 hours (the full LP with all special constraints inactive), but on the other hand, the full LP model takes more than 500.000 seconds for those eight instances. Using special constraints as cuts will only affect approximately one third of the instances, and the effect is questionable. Therefore, we will not consider the special constraints to be used as cuts.

For the student sectioning group, we see very long solution times. However, the number of iterations performed was huge, and most of them were performed at the full LP objective. The student sectioning constraints are one of the largest parts of the model, which means that when performing cuts, much time is saved by not generating the complete set of constraints. On the background of the preliminary tests, it seems the student sectioning is the most promising group for cuts. We should keep in mind that six of the 30 instances do not consider student sectioning, which means they can never benefit from this type of cut. The six instances are relatively easier to solve compared to the ones with student sectioning. Three of the six instances are solved in less than a minute, and the longest takes around an hour. Since the student sectioning cut procedure was the most promising in the preliminary tests, we will use this for the rest of the report.

6 The cuts

The preliminary tests show that the student sectioning group is the most promising constraint group for a cutting algorithm. The student sectioning group contains constraints for handling students' class assignments and penalizing student conflicts. The primary variable for the student sectioning constraints is the binary $e_{s,c}$ variable, which is set to 1 if student s is assigned class c. The constraints for assigning the students to the courses correctly (must attend one class for all subparts in one of the course configurations) are all equality constraints. Thus they are not investigated for cutting. The remaining student sectioning constraints consider the limitation on the number of students attending a class and the parent-child relationship between classes. The student conflict constraints consist of two overlap constraints; one for time overlap and one for time-room overlap of class pairs. The constraints set the auxiliary binary variable o_{c_i,c_j} to 1 if two classes overlap, and 0 otherwise. Student conflict variable constraints are used to set the auxiliary binary variable χ_{s,c_i,c_j} to 1 if a student is attending two classes with an overlap variable equal to 1. Finally, the star constraints are used in cases where the number of attendees of a class pair is always fixed, making the conflict penalty solely related to the overlap variable of the class pair. The different constraints can be seen in Table 9.

Constraint type	Description
Class limit	Limits the number of students assigned a class
Parent-child relation	Sets the student to attend the parent class if the child class is attended
Class star conflicts	For class pairs with a fixed number of equal attendees, a star cover is used to penalize overlaps
Class time overlap	Sets the overlap o_{c_i,c_j} variable if a pair of classes is overlapping
Class time-room overlap	Sets the overlap variable o_{c_i,c_j} if a pair of classes is scheduled on the same day in non-overlapping times, but the time between the classes is less than the travel time between the assigned rooms
Conflict variable LB	Sets the student conflict binary variable χ_{s,c_i,c_j} if a pair of classes overlap and a student is attending both classes

Table 9 Description of the student sectioning constraints.

The constraints of Table 9 are the ones that we will consider for the cutting plane algorithm. The star constraints are related to the specific star cover of the conflict graph, which means that changes to the conflict graph may result in another cover and thereby another model formulation. By this fact, we cannot detect an overlap of those classes without generating the star covers because we do not know what defines an overlap since that is defined from the star cover. We can either leave star covers out of the cutting procedure by having all in the relaxed LP model or pre-generate the star covers separately from the LP model in a list to check for violations when cutting (similar to lazy constraints).

Even with these two options, there are different ways to add the cuts. We will test some of them to see if we can find which performs better. The general cutting procedure follows the structure of Algorithm 2 and we will consider different procedures to find the cutting constraints GetCuttingConstraints(sol^*, tol, minLP). The cut procedure will take an optimal minLP solution and a tolerance (0.01). The cut procedure will then look for infeasibilities (student conflicts not accounted for) in the solution and add them to the minLP. Tolerance is used when checking for violations.

Algorithm 2 Cutting plane algorithm

```
1: minLP = generate LP model of the problem without the
   cutting constraints
2: repeat
3:     sol* = solve minLP
4:     GetCuttingConstraints(sol*, tol, minLP)
5: until no more constraints are added
```

First, we will consider the minLP model containing the star cover constraints, which means we cut on the class limit, parent-child relation, class overlapping, and conflict constraints. The class limit constraints are simply a check-and-add procedure. A constraint is generated for each class where the sum of attending students is larger than the class limit. The same goes for the parent-child relationship. Generate the constraints for each student attending a child class but not the parent class. The pseudo-code is shown in Algorithm 3 and 4 and they will be used in all the following cutting methods.

To get the student conflict cuts, we must find the overlapping classes (both time and time-room overlaps) and the class pairs attended by the same students. This part might perform depending on the implementation. Should we check the student assignments before the

Algorithm 3 Class limit cuts

procedure CutClassLimit(sol^*, tol, minLP)
input: sol^* = a LP solution, tol = the tolerance, minLP = the LP
for all $c \in \mathcal{C}$ **do**
  if $\sum_{s \in \mathcal{S}_c} e_{s,c} > c^{\text{limit}} + tol$ **then**
    add the class limit constraint to minLP

Algorithm 4 Parent-child relation cuts

procedure CutParentChild(sol^*, tol, minLP)
input: sol^* = a LP solution, tol = the tolerance, minLP = the LP
for all $c \in \mathcal{C} : (c^{\text{parent}} \neq \text{null})$ **do**
  for all $s \in \mathcal{S}_c$ **do**
    if $e_{s,c} > e_s^{c^{\text{parent}}} + tol$ **then**
      add the parent-child rel. constraint to minLP

overlap or the other way around, and how many constraints should be added when a violation is found. We know there is much symmetry in the student sectioning part, which means that the number of iterations could be reduced if more constraints can be deducted in each iteration. On the other hand, adding too many redundant constraints might lead to slower solution time.

The order will be the base for our first test. We create two cutting procedures; CutOverlapFirst and CutStudentFirst (Algorithm 5 and 6). Algorithm 5 presents a cutting procedure that checks for overlaps between class pairs. When a time or a time-room overlap is found, the respective constraints are added. Afterward, the students are checked. When a student is attending two overlapping classes, the conflict variable LB constraints are added.
Algorithm 6 checks the students first. When a student attends two overlapping classes, the conflict variable LB constraints are added. When a student attends a non-overlapping class pair, the pair is saved in a list to be checked later. When all students have been checked, we check the class pairs. Again the overlap is divided into time and time-room overlaps. The corresponding constraints are added if an overlap is found.

The cutting plane algorithm (Algorithm 2) has been run with the overlap first (5) and student first (Algorithm 6) once for each instance to test the performance against the full LP. We do not want to compare the solver time for these cutting tests as the cutting methods have the advantage that they save time by not generation the full model. Instead, we compare the complete run time in Table 10. We have used an average of five runs with a seven day time limit for the full LP. The cutting procedures are allowed 24 hours Gurobi solve time in each iteration and seven days total run time. Some runs showed to take a long time for the full LP. Those instances will be given a seven-day Gurobi time limit and 21 days total time limit. Table 10 shows that sometimes CutStudentFirst is faster than the full LP, and for two instances, the cutting procedure finishes where the full LP was not able to solve within seven days. When comparing the CutOverlapFirst and CutStudentFirst, we find that CutStudentFirst is faster on all but one instance, while generating fewer cuts and in most of the instances solves more iterations. We also present some plots of some of the instances. In Figure 6 we also see two cases that were not solved by the full LP but can be solved by both cutting methods. In Figure 6 we see two cases where the full LP was solved. In the case of *pu-d9-fal19*, CutOverlapFirst does not terminate within the time limit, nor does it reach the full LP objective. CutStudentFirst reaches the LP objective before the full LP but does not terminate until after the full LP solution time. It is general that CutStudentFirst reaches the LP objective first and finishes before CutOverlapFirst. By this conclusion, we will continue with the student first approach.

Algorithm 5 Overlap first

procedure CutOverlapFirst(sol^*, tol, minLP)
input: sol^* = a LP solution, tol = the tolerance, minLP = the LP
CutClassLimit(sol^*, tol, minLP) ▷ add class limit cuts
CutParentChild(sol^*, tol, minLP) ▷ add parent-child relation cuts
for all $(c_i, c_j) \in \mathcal{C}$ **do** ▷ for all class pairs
  if (c_i, c_j) is violating a time overlap constraint **then**
    add the class time overlap constraint to minLP
  if (c_i, c_j) is violating a time-room overlap constraint **then**
    add the class time-room overlap constraint to minLP
for all $s \in \mathcal{S}$ **do** ▷ for all students
  for all $(c_i, c_j) \in \mathcal{C}_s$ **do** ▷ and for all possible class pairs
    if A student conflict constraint is violated **then**
      add the conflict variable LB constraints to minLP

Algorithm 6 Student first

procedure CutStudentFirst(sol^*, tol, minLP)
input: sol^* = a LP solution, tol = the tolerance, minLP = the LP
CutClassLimit(sol^*, tol, minLP) ▷ add class limit cuts
CutParentChild(sol^*, tol, minLP) ▷ add parent-child relation cuts
ol = set of overlapping class pairs ▷ initially empty
for all $s \in \mathcal{S}$ **do** ▷ for all students
  for all $(c_i, c_j) \in \mathcal{C}_s$ **do** ▷ and for all possible class pairs
    if Student attending both classes c_i and c_j **then**
      add (c_i, c_j) to ol
      if a student conflict exists **then** ▷ if student attends both classes and the classes overlap
        add the conflict variable LB constraints to minLP
for all $(c_i, c_j) \in ol$ **do** ▷ for all the discovered overlapping class pairs
  if (c_i, c_j) is violating a time overlap constraint **then**
    add the class time overlap constraint to minLP
  if (c_i, c_j) is violating a time-room overlap constraint **then**
    add the class time-room overlap constraint to minLP

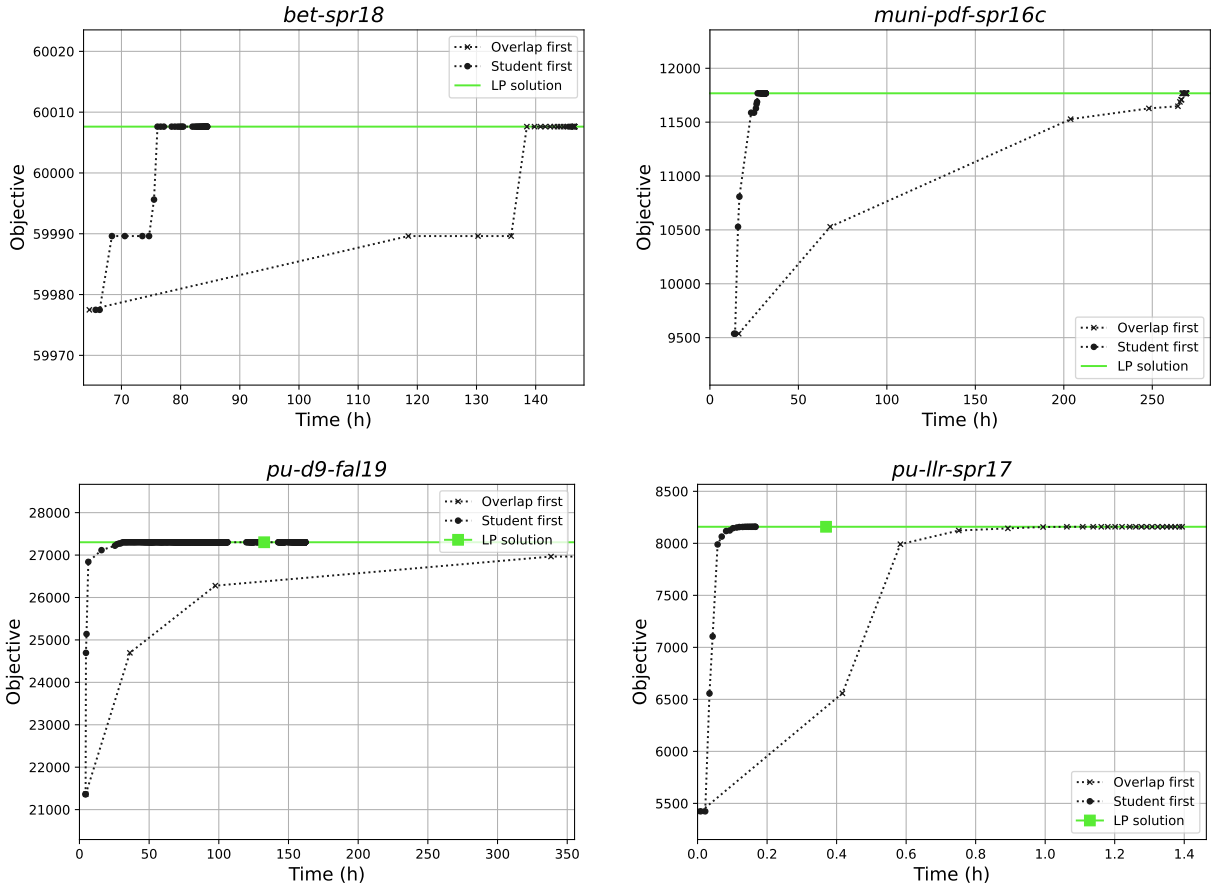

Fig. 6 The two instances, *bet-spr18* and *muni-pdf-spr16c*, not solved by the full LP model in 7 days. Both instances was solved in the preliminary test, which gives the LP objective. Two instances both solved by the student first cuts. *Pu-d9-fal19* was not solved by the overlap first method. In the finished cases we see the tail along LP objective. There is a clear difference between the two methods.

	Full LP	CutOverlapFirst			CutStudentFirst		
Instance	Total time	Total time	Solver count	Number of cuts	Total time	Solver count	Number of cuts
agh-fis-spr17	16,691	218,438	28	405,995	**13,859**	56	32,165
agh-ggis-spr17	**314**	4,267	22	152,957	516	18	29,585
bet-fal17		1,567,638	14	171,481	**722,237**	30	27,239
iku-fal17	-	-	-	-	-	-	-
mary-spr17	**524**	3,500	12	163,273	547	36	9,760
muni-fi-spr16	864	631	20	61,749	**257**	25	16,488
muni-fsps-spr17	**43**	455	13	59,745	81	22	6,845
muni-pdf-spr16c		969,646	21	236,767	**113,543**	37	37,580
pu-llr-spr17	1,331	4,446	18	158,253	**600**	20	34,365
tg-fal17	-	-	-	-	-	-	-
agh-ggos-spr17	17,173	60,012	16	197,169	**7,221**	32	20,627
agh-h-spr17	**53,737**	563,476	23	72,388	121,413	16	4,366
lums-spr18	-	-	-	-	-	-	-
muni-fi-spr17	**379**	651	20	52,592	452	30	17,122
muni-fsps-spr17c	**3,058**	588,702	132	440,384	555,368	230	135,782
muni-pdf-spr16	**536,523**	time limit[1]			time limit[1]		
nbi-spr18	326	3,520	14	100,238	329	11	5,298
pu-d5-spr17	**773**	37,726	155	234,599	46,859	230	321,021
pu-proj-fal19		time limit[2]			time limit[2]		
yach-fal17	**321**	1,559	18	71,495	539	36	19,250
agh-fal17		time limit			time limit		
bet-spr18		527,706	27	201,653	**304,139**	37	34,346
iku-spr18	-	-	-	-	-	-	-
lums-fal17	-	-	-	-	-	-	-
mary-fal18	**2,461**	7,072	16	203,876	2,975	37	32,798
muni-fi-fal17	**488**	799	15	57,474	654	29	20,038
muni-fspsx-fal17	**12,716**	time limit[1]			time limit[1]		
muni-pdfx-fal17		time limit			time limit		
pu-d9-fal19	**477,168**	time limit[2]			583,472	393	344,779
tg-spr18	-	-	-	-	-	-	-

Table 10 Comparison of the total run time of the full LP, the overlap first and student first cutting methods. *Time limit* indicates that the Gurobi time limit of 7 days was reached. *Time limit*[1] indicates that the total run time limit of 7 days was reached. *Time limit*[2] indicates that the total run time limit of 21 days was reached.

Since we have shown that the student first procedure should be the one to consider, we now introduce a more aggressive cutting method where we add cuts for all possible students when a conflict is found. The idea is that it should limit the symmetry when two students can attend the same class pairs. We do this to reduce the number of iterations where similar students are interchanged in the same classes. The new CutAllStudents cutting method is shown in Algorithm 7.

From Table 11 we see that, as expected, the number of cuts used is increased when adding cuts for all the students when a conflict happens. We would also expect that the number of iterations would be lower since we would have found more cuts in each iteration, and thereby the earlier iterations might contain the cuts from the later ones. Even though this is the case in most instances, we also see the opposite. Regarding the running time, there is no consistency over the instances. We will have to examine the plots of the runs. The plots are shown in Figure 7 and the overall picture is that the CutAllStudents method converges towards the LP objective slower than the CutStudentFirst method. Again, there are some instances where it is the other way around.

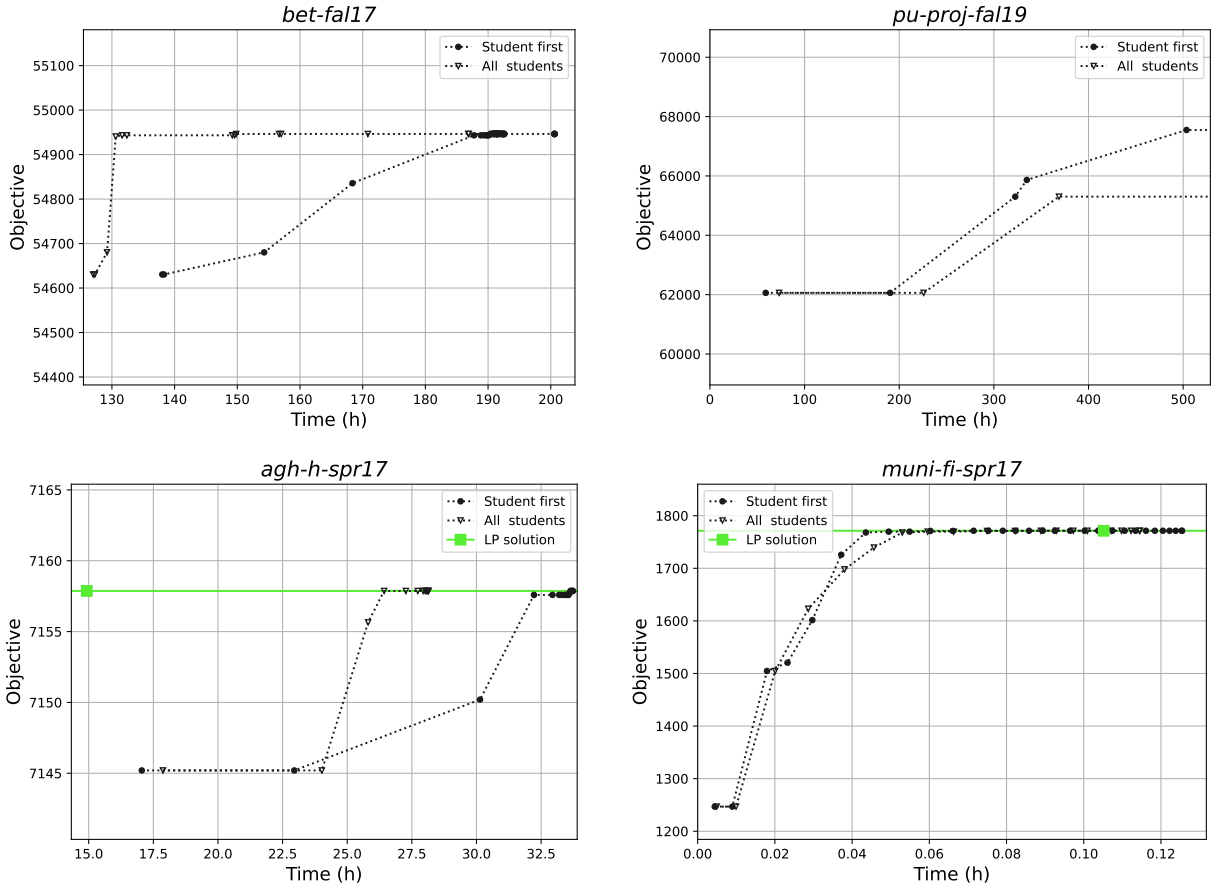

Fig. 7 Some examples of cutting procedures student first and all students. The figures shows a how the procedures are varying in the different instances. *Bet-fal17* and *pu-proj-fal19* was not able to solve the full LP.

Algorithm 7 All students

procedure CutAllStudents(sol^*, tol, minLP)
input: sol^* = a LP solution, tol = the tolerance, minLP = the LP
CutClassLimit(sol^*, tol, minLP) ▷ add class limit cuts
CutParentChild(sol^*, tol, minLP) ▷ add parent-child relation cuts
ol = set of overlapping class pairs ▷ initially empty
for all $s \in \mathcal{S}$ **do** ▷ for all students
  for all $(c_i, c_j) \in \mathcal{C}_s$ **do** ▷ and for all possible class pairs
    if Student attending both classes c_i and c_j **then**
      add (c_i, c_j) to ol
      if a student conflict exists **then** ▷ if student attends both classes and the classes overlap
        for all students able to attend c_i and c_j **do** ▷ all students eligible for both classes
          add the *conflict variable LB*'s to minLP
for all $(c_i, c_j) \in ol$ **do** ▷ for all the discovered overlapping class pairs
  if (c_i, c_j) is time overlapping **then** ▷ if the class pair is time overlapping
    add the class time overlap constraint to minLP ▷ add the relevant constraints
  if (c_i, c_j) is time-room overlapping **then** ▷ if the class pair is time-room overlapping
    add the class time-room overlap constraint to minLP ▷ add the relevant constraints

	Full LP	CutStudentFirst			CutAllStudents		
Instance	Total time	Total time	Solver count	Number of cuts	Total time	Solver count	Number of cuts
agh-fis-spr17	16.691	13.859	56	32.165	**7.738**	38	146.851
agh-ggis-spr17	**314**	516	18	29.585	559	19	120.519
bet-fal17		722.237	30	27.239	**689.599**	25	111.542
iku-fal17	-	-	-	-	-	-	-
mary-spr17	**524**	547	36	9.760	1.032	54	27.352
muni-fi-spr16	864	**257**	25	16.488	394	25	107.216
muni-fsps-spr17	**43**	81	22	6.845	86	21	31.468
muni-pdf-spr16c		113.543	37	37.580	**104.992**	37	66.550
pu-llr-spr17	1.331	**600**	20	34.365	1.137	20	275.542
tg-fal17	-	-	-	-	-	-	-
agh-ggos-spr17	17.173	**7.221**	32	20.627	9.668	39	68.240
agh-h-spr17	**53.737**	121.413	16	4.366	101.318	11	8.568
lums-spr18	-	-	-	-	-	-	-
muni-fi-spr17	**379**	452	30	17.122	412	20	112.790
muni-fsps-spr17c	**3.058**	555.368	230	135.782	643.726	148	207.283
muni-pdf-spr16	**536.523**	time limit[1]			time limit[2]		
nbi-spr18	326	329	11	5.298	347	10	5.569
pu-d5-spr17	**773**	46.859	230	321.021	62.989	151	424.177
pu-proj-fal19		time limit[2]			time limit[2]		
yach-Fal17	**321**	539	36	19.250	412	24	48.682
agh-fal17		time limit			time limit		
bet-spr18		304.139	37	34.346	**290.448**	30	91.700
iku-spr18	-	-	-	-	-	-	-
lums-fal17	-	-	-	-	-	-	-
mary-fal18	2.461	2.975	37	32.798	4.538	42	167.037
muni-fi-fal17	488	654	29	20.038	855	27	205.152
muni-fspsx-fal17	12.716	time limit[1]			time limit[2]		
muni-pdfx-fal17		time limit			time limit		
pu-d9-fal19	477.168	583.472	393	344.779	896.787	268	1.737.362
tg-spr18	-	-	-	-	-	-	-

Table 11 Comparison of the total run time of the full LP, the student first and all students cutting methods. *Time limit* indicates that the Gurobi time limit of 24 hours was reached. *Time limit*[1] indicates that the total run time limit of 7 days was reached. *Time limit*[2] indicates that the total run time limit of 21 days was reached.

The previous cutting methods have considered LP models with the class star conflict constraints included in the minLP model. We will now investigate the effect of having those implemented as cuts. The problem with the star conflict constraints is that they rely on the star cover of a conflict graph, which means that we must examine the same conflict graphs in the cutting procedure used to generate the star cover in the full LP. Otherwise, we end up comparing different models. As we have to generate the star covers before solving, we do not save the generation time of those constraints. They will be a kind of lazy constraints. We present two versions of adding the star conflicts to the CutStudentFirst procedure; one where only the violated star constraint is added and another where the whole star cover containing the violated star is added. The procedures are shown in Algorithm 8 and 9. Again we see a very diverse picture when looking at the plots. Some representative plots are shown in Figure 8. There is no consistent method of the three faster than the others.

Algorithm 8 Student first - Including stars

procedure CutStudentFirstIncludeStars(sol^*, tol, minLP, $\mathtt{S}$)
input: sol^* = a LP solution, tol = the tolerance, minLP = the LP, $\mathtt{S}$ = pre-generated list of star constraints
1: **for all** $\mathtt{s} \in \mathtt{S}$ **do** ▷ for all star constraints
2: **if** $\mathtt{s}$ is violated **then** ▷ if the constraint is violated
3: add $\mathtt{s}$ to minLP ▷ add the star constraint to minLP
4: CutStudentFirst(sol^*, tol, minLP)

Algorithm 9 Student first - Including stars (all)

procedure CutStudentFirstIncludeStarsAll(sol^*, tol, minLP, $\mathtt{S}_G$)
input: sol^* = a LP solution, tol = the tolerance, minLP = the LP, $\mathtt{S}_G$ = pre-generated list of star constraints grouped by conflict graph G
1: **for all** $g \in G$ **do** ▷ for all conflict graphs
2: **if** any $\mathtt{s} \in \mathtt{S}_G(g)$ is violated **then** ▷ if any star constraint of g is violated
3: add all $\mathtt{S}_G(g)$ to minLP ▷ add all star constraints of g to minLP
4: CutStudentFirst(sol^*, tol, minLP)

	Full LP	CutStudentFirst			CutStudentFirstIncludeStars			CutStudentFirstIncludeStarsAll		
Instance	Total time	Total time	Solver count	Number of cuts	Total time	Solver count	Number of cuts	Total time	Solver count	Number of cuts
agh-fis-spr17	16,691	13,859	56	32,165	**9,097**	29	25,077	10,726	40	29,012
agh-ggis-spr17	**314**	516	18	29,585	740	23	33,183	756	18	30,692
bet-fal17		722,237	30	27,239	**576,223**	32	30,401	685,833	19	29,315
iku-fal17	-	-	-	-	-	-	-	-	-	-
mary-spr17	**524**	547	36	9,760	1,099	65	12,183	566	32	8,916
muni-fi-spr16	864	**257**	25	16,488	454	30	21,728	458	30	20,681
muni-fsps-spr17	**43**	81	22	6,845	89	22	7,123	90	22	6,680
muni-pdf-spr16c		113,543	37	37,580	51,792	31	29,328	**48,576**	30	29,463
pu-llr-spr17	1,331	**600**	20	34,365	1,199	29	40,103	1,029	24	39,173
tg-fal17	-	-	-	-	-	-	-	-	-	-
agh-ggos-spr17	17,173	**7,221**	32	20,627	8,181	24	20,317	10,734	23	20,870
agh-h-spr17	53,737	121,413	16	4,366	**35,288**	16	4,963	81,768	19	6,432
lums-spr18	-	-	-	-	-	-	-	-	-	-
muni-fi-spr17	**379**	452	30	17,122	763	33	22,838	616	31	20,372
muni-fsps-spr17c	**3,058**	555,368	230	135,782	24,108	73	50,343	63,206	135	78,235
muni-pdf-spr16	**536,523**	time limit[1]			time limit[2]			924,765	38	45,476
nbi-spr18	**326**	329	11	5,298	938	27	6,646	879	17	5,368
pu-d5-spr17	**773**	46,859	230	321,021	24,871	258	156,052	27,058	267	159,258
pu-proj-fal19		time limit[2]			time limit[2]			time limit[2]		
yach-fal17	**321**	539	36	19,250	531	25	15,368	478	24	14,042
agh-fal17		time limit			Numeric error			Numeric error		
bet-spr18		**304,139**	37	34,346	501,106	44	33,838	434,031	30	33,482
iku-spr18	-	-	-	-	-	-	-	-	-	-
lums-fal17	-	-	-	-	-	-	-	-	-	-
mary-fal18	**2,461**	2,975	37	32,798	6,099	69	42,737	6,839	84	47,078
muni-fi-fal17	**488**	654	29	20,038	1,024	31	23,052	1,145	30	21,231
muni-fspsx-fal17	**12,716**	time limit[1]			time limit[2]			time limit[2]		
muni-pdfx-fal17		time limit			time limit[2]			time limit[2]		
pu-d9-fal19	477,168	583,472	393	344,779	582,552	315	302,324	**286,558**	225	267,439
tg-spr18	-	-	-	-	-	-	-	-	-	-

Table 12 Comparison of the total run time of the full LP, the student first and the two cutting methods including the star constraints. *Time limit* indicates that the Gurobi time limit of 7 days was reached. *Time limit*[1] indicates that the total run time limit of 7 days was reached. *Time limit*[2] indicates that the total run time limit of 21 days was reached.

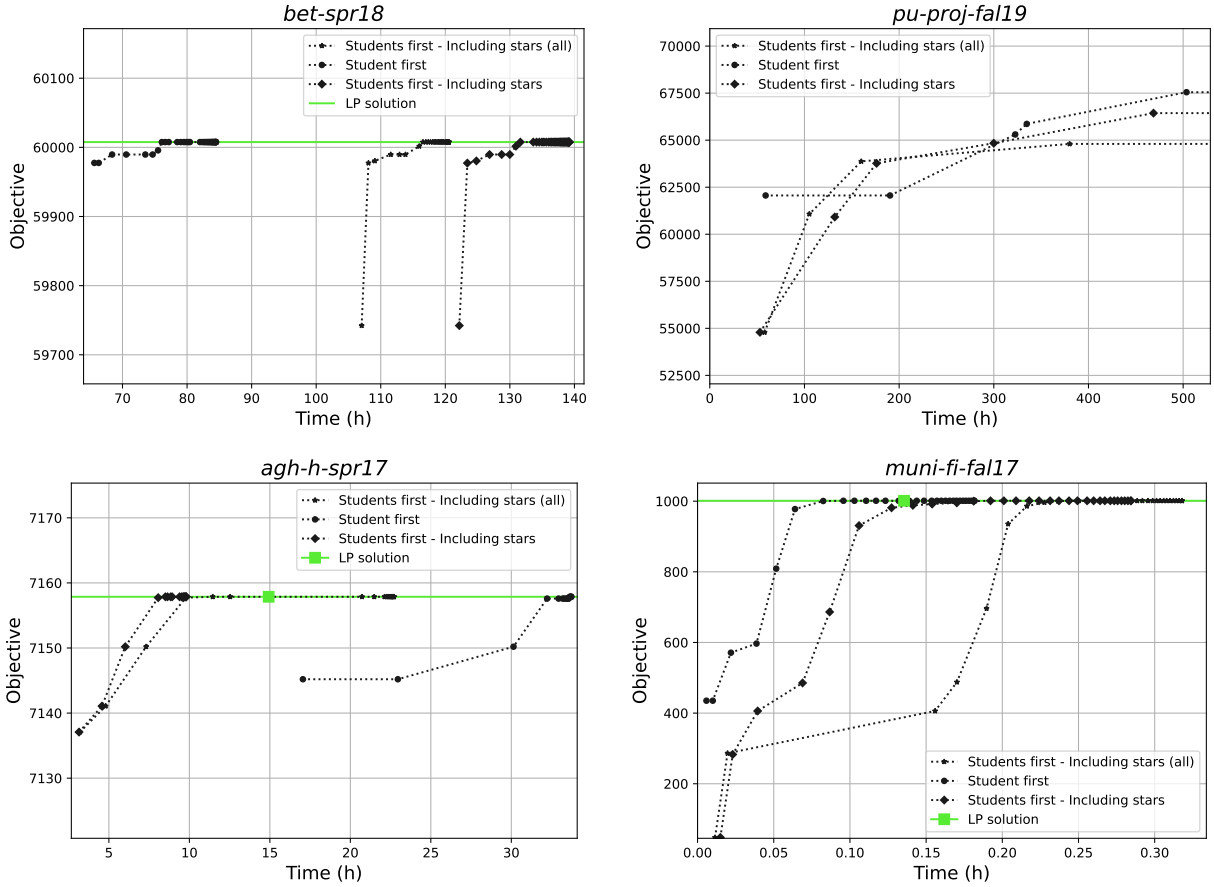

Fig. 8 Some examples of cutting procedures Student first, including stars and including stars (all). The figures shows a how the procedures are varying in the different instances. In some instances one is better than the other and other instances it is the other way around. *bet-spr18* and *pu-proj-fal19* was not able to solve the full LP, however the preliminary tests found the LP solution to *bet-spr18*.

7 Results

The preliminary tests showed that it was clear which constraints were binding in the LP solution. The tests also showed that solving the LP with only the binding student sectioning constraints was favorable over solving the full LP. There is a slight favor in having only the binding graph constraints for the instances that do not consider students, which might be because those instances use the `SameAttendees` distribution constraints to model curriculum overlap. Having only the binding special constraints benefits some instances and weakens others. By the conclusion of the preliminary tests, we ruled out the SameRoom group for further consideration. In the next step, we tried an iterative implementation of activating constraints as they were violated by solutions to a relaxed LP model. The results showed that the SameAttendees group and room double booking group used few iterations and added few constraints. To improve those methods, we would have to know the set of binding constraints before solving the LP, which is impossible. The special constraints group is only applicable for around one third of the instances and around another third reached the Gurobi time limit of seven days, which means that at most one third of the instances could possibly benefit from this method. The graph constraints showed to be somewhat effective, but as the student sectioning group showed better performance, we chose to continue with that.

The different student sectioning cutting procedures show that it was better to look at the student assignments before the overlapping classes to find the cuts to add. We tried different versions of the cutting technique to add more or fewer cuts when a violation was found. The results for those was a slight tilt towards adding fewer cuts in each iteration. As for the star conflict constraints, some instances would benefit from having those in the LP, and others would benefit from adding them as cuts. To include all instances, it could be interesting to look into a combination of graph and student conflict cuts. The results also show that the cutting procedure often reaches the LP solution objective before the full LP is solved, which might be useful in a branch-and-cut setting. Furthermore, it should also be looked into if it can be foreseen which cut method works on an instance from the instance data characteristics.

This report has shown improved lower bounds for two instances; *pu-proj-fal19* and *muni-pdfx-fal17*.

Instance	Best solution	Best LB	New LB
pu-proj-fal19	117,425	54,872	67,549
muni-pdfx-fal17	84,703	26,711	28,057

Table 13 Table showing the instances with new lower bounds, the old lower bounds (`www.dsumsoftware.com/itc2019`) and the best solution value (`www.itc2019.org/results`). The new LB for *pu-proj-fal19* was generated by CutStudentFirst. The new LB for *muni-pdfx-fal17* was generated by CutStudentFirstIncludeStars.

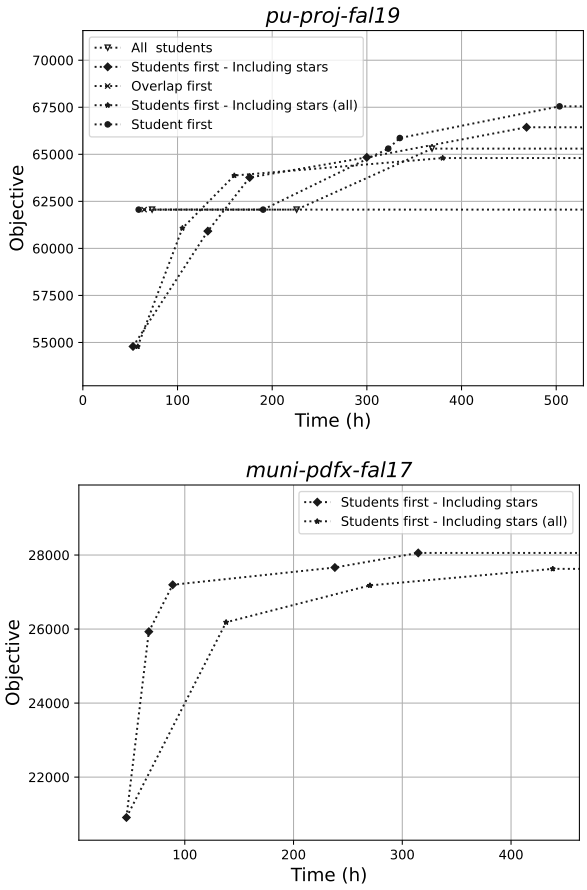

Fig. 9 The plots of the two instances with new lower bounds and all the cutting methods that solved at least once.

References

Burke, E. K., Mareček, J., Parkes, A. J., and Rudová, H. (2012). A branch-and-cut procedure for the Udine Course Timetabling problem. *Annals of Operations Research*, 194(1):71–87. https://doi.org/10.1007/s10479-010-0828-5.

Dueck, G. (1993). New optimization heuristics: The great deluge algorithm and the record-to-record travel. *Journal of Computational Physics*, 104(1):86–92. https://doi.org/10.1006/jcph.1993.1010.

Er-rhaimini, K. (2019). Forest growth optimization for solving timetabling problems. Available at https://www.itc2019.org/papers/itc2019-er-rhaimini.pdf. Accessed: 2021-07-13.

Gashi, E. and Sylejmani, K. (2019). Simulated annealing with penalization for university course timetabling. Available at https://www.itc2019.org/papers/itc2019-gashi.pdf. Accessed: 2021-07-13.

Holm, D., Mikkelsen, R., Sørensen, M., and Stidsen, T. (2022). A graph-based MIP formulation of the International Timetabling Competition 2019. *Journal of Scheduling*. https://doi.org/10.1007/s10951-022-00724-y.

Lemos, A., Monteiro, P. T., and Lynce, I. (2019). Itc-2019: A maxsat approach to solve university timetabling problems. Available at https://www.itc2019.org/papers/itc2019-lemos.pdf. Accessed: 2021-07-13.

Mikkelsen, R. and Holm, D. (2022). A parallelized matheuristic for the International Timetabling Competition 2019. *Journal of Scheduling*. https://doi.org/10.1007/s10951-022-00728-8.

Müller, T. (2020). Itc 2019: Preliminary results using the unitime solver. Available at https://www.unitime.org/papers/itc2019-patat2020.pdf. Accessed: 2021-07-13.

Rappos, E., Thiémard, E., Robert, S., and Hêche, J.-F. (2019). International Timetabling Competition 2019: A Mixed Integer Programming Approach for Solving University Timetabling Problems. Available at https://www.itc2019.org/papers/itc2019-rappos.pdf. Accessed: 2021-07-13.

Appendices

A Binding constraints

Table 14 shows the proportion of binding constraints for each of the cut constraint groups of each instance.

Instance	RoomDoublebook	Graph	SameAttendees	SameRoom	Special	Student sectioning
agh-fis-spr17	0.33%	5.23%	0.02%	100%	69.38%	6.10%
agh-ggis-spr17	3.29%	36.03%	50.00%	100%	64.56%	11.34%
bet-fal17						
iku-fal17	3.46%	21.18%	3.48%	99.81%	-	-
mary-spr17	5.80%	12.56%	7.92%	100%	-	3.61%
muni-fi-spr16	6.78%	8.22%	-	100%	63.38%	2.45%
muni-fsps-spr17	3.58%	16.96%	0.53%	100%	-	6.39%
muni-pdf-spr16c						
pu-llr-spr17	3.71%	30.07%	1.58%	100%	60.29%	7.69%
tg-fal17	2.28%	30.36%	100%	-	-	-
agh-ggos-spr17	0.80%	15.48%	0.01%	100%	75.93%	6.19%
agh-h-spr17	0.07%	10.83%	0.01%	76.74%	70.09%	5.57%
lums-spr18	1.58%	2.41%	0.03%	-	-	-
muni-fi-spr17	6.37%	6.61%		100%	71.74%	2.13%
muni-fsps-spr17c	2.92%	5.01%	0.00%	100%	-	2.78%
muni-pdf-spr16	3.11%	8.45%	0.38%	100%	78.36%	3.19%
nbi-spr18	7.35%	11.28%	0.93%	-	-	5.25%
pu-d5-spr17	20.20%	4.01%	-	100%	69.26%	5.81%
pu-proj-fal19						
yach-fal17	4.19%	4.47%	0.07%	100%	1.09%	4.55%
agh-fal17						
bet-spr18						
iku-spr18	3.61%	18.99%	0.64%	100%	-	-
lums-fal17	1.91%	2.90%	0.00%	-	-	-
mary-fal18	6.65%	8.81%	1.45%	100%	-	3.87%
muni-fi-fal17	12.01%	8.68%	-	100%	75.15%	1.64%
muni-fspsx-fal17	5.15%	6.01%	0.03%	98.57%	-	2.11%
muni-pdfx-fal17						
pu-d9-fal19	3.27%	9.91%	0.70%	100%	66.63%	3.64%
tg-spr18	2.11%	9.54%	100%	100%	14.00%	-
Average	4.61%	12.25%	13.39%	98.76%	59.99%	4.68%

Table 14 The proportions of binding constraints in the different instances. The dash (-) means that there are no constraints of the group.