International Timetabling Competition 2019: A Mixed Integer Programming Approach for Solving University Timetabling Problems

Efstratios Rappos · Eric Thiémard · Stephan Robert · Jean-François Hêche

Abstract This summary article presents the mathematical programming approach used to solve and optimize the problem instances of the International Timetabling Competition 2019. The optimization problem was modeled as a mixed integer program which was solved using traditional branch-and-cut methods. Several innovative elements enabled to achieve good performance, such as the precalculation of several characteristics of the instances, the aggregation of constraints and the efficient use of auxiliary variables in the formulation. The computational implementation consisted of a first stage algorithm to obtain a feasible solution and an iterative local search metaheuristic to improve the quality of the resulting timetable. The solutions produced using this algorithm resulted in a ranking of second place in the competition.

Keywords ITC 2019 · timetabling problems · integer programming · combinatorial optimization

1 Introduction

This extended abstract describes the method used to solve the timetabling problems of the 2019 International Timetabling Competition [1]. Overall, we were able to solve 29 out of the 30 problem instances; the instance "agh-fal17" was not solved in time for the end of the competition. The mathematical modeling approach formulated the problem as a linear mixed integer program (MIP) and used techniques from large neighborhood search [2],[3],[4] and metaheuristics [5] to improve the solution quality.

2 Model formulation

The MIP formulation uses four sets of binary 0-1 variables, x, y, z and Z, representing the class times, class rooms, student-class allocation and student-course configuration alloca-

Efstratios Rappos · Eric Thiémard · Stephan Robert · Jean-François Hêche
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud (HEIG-VD)
University of Applied Sciences of Western Switzerland (HES-SO)
Yverdon-les-Bains, Switzerland
E-mail: efstratios.rappos@heig-vd.ch

tion respectively. The indices of these variables are described in Table 1 and their values uniquely represent a solution to a timetabling problem.

Variable	*Value*
$x_{c,t}$	1 if the class c takes place at time t, 0 otherwise
$y_{c,r}$	1 if the class c takes place in room r, 0 otherwise
$z_{s,c}$	1 if the student s is assigned to the class c, 0 otherwise
$Z_{s,f}$	1 if the student s follows the course configuration f, 0 otherwise

Table 1 Variables used in the formulation

The binary variables are linked by a set of linear constraints representing the solution requirements of the timetabling problems of the 2019 International Timetabling Competition. The hard constraints must be satisfied by any feasible solution and are summarized in Table 2.

Constraint	*Meaning*
C-1	Every class must be assigned a time
C-2	Every class must be assigned a room, where applicable
C-3	Every student must attend exactly one class from each subpart of the selected course configuration for each course that he must attend
C-4	For two classes with a parent-child relationship, if a class is assigned to a student then the parent class must also be assigned
C-5	Every student must be assigned a course configuration for every course that he follows
C-6	The capacity of each class in terms of the number of students must be satisfied
C-7	A room cannot be used when it is unavailable
C-8	Two classes cannot take place at the same time in the same room
C-9	Any hard distribution constraints must be satisfied

Table 2 Constraints used in the formulation

The first seven constraints are straightforward to formulate as linear inequalities using the above decision variables. For example, C-1 is represented by the formula $\sum_t x_{c,t} = 1$ for each class c; C-3 is expressed as $\sum_c z_{s,c} = Z_{s,f}$ for each student s, where the sum is over all classes c of one subpart of the course configuration f, and C-5 is $\sum_f Z_{s,f} = 1$ for every student s. The last two constraints C-8 and C-9 are modeled as inequalities of the form:

$$x_{c_1,t_1} + y_{c_1,r_1} + x_{c_2,t_2} + y_{c_2,r_2} \leq 3 \tag{1}$$

for every combination of class times and rooms which leads to a violated constraint. The inequality (1) prevents all four terms from taking the value 1 and therefore disallows this specific combination. There are however four types of hard constraints (namely, the special MaxDays, MaxDayload, MaxBreaks and MaxBlock constraints) that cannot be represented by inequalities of the form (1). This is because these constraints cannot be expressed in the form "for each pair of classes"; for these constraints an additional step is taken. Each time a potential new solution is found we need to check that these special constraints are satisfied, and if they are not, reject the solution by adding the appropriate inequalities, similar to (1) but with more terms, that forbid this combination of variables.

The objective function consists of four linear terms which correspond to the four solution quality criteria of the competition [1], namely the class time and room assignment costs, soft distribution constraint costs and student conflicts. The first two terms are simply

the weighted sum of the x and y variables. For the costs associated with the soft distribution constraints, an auxiliary variable is introduced in (1) which specifies if the constraint is satisfied or not, and the sum of these variables is used as the third term of the objective function. A similar method is used for the student conflicts. An auxiliary variable is introduced for every student and pair of classes he may follow, specifying whether a conflict exists or not. However, as the number of potential student conflicts can be very large, we aggregate the penalties associated with every pair of classes into one equation (for all students) by introducing an indicator variable which specifies whether a student follows both classes or not. Note that we are able to take into account both types of student conflict: where two classes overlap and when the travel time between the classes is insufficient.

Several innovative techniques are used to deal with the very large number of constraints of the above formulation, which makes it possible to solve the competition instances in a reasonable amount of time. The efficiency optimizations include:

- Extensive use of precalculated problem characteristics to save time between runs, for example the minimum and maximum gap and travel distances between classes
- Four types of logical checks to eliminate variables whose value can be deduced
- Removal of constraints that are always satisfied
- Reduction of the problem size by constraint aggregation

As an example of a logical check, we examine the existence of two classes which must take place in the same room. If one of these classes has its time assignment fixed, we can then exclude those time assignments of the second class which produce a conflict.

The computational implementation was done in Java using the commercial software CPLEX and Gurobi as the mixed integer programming solvers. The solution strategy is outlined in Agorithm 1 and consisted of two stages: the first stage focused at obtaining a feasible solution via an incremental addition of the hard constraints, whereas the second stage is a metaheuristic which combines iterative local search with mixed integer programming and aims to improve the solution quality.

The first stage performs a progressive addition of constraints into the model using slack variables to account for any violated constraints. Once a feasible solution is obtained the second stage iteratively improves the solution quality. In the end, the algorithm will produce better and better solutions until the optimization is stopped for practical reasons.

In both stages a number of decision variables is fixed to their last-solution values to reduce the size of the MIP. Several strategies of fixing the variables were developed and implemented sequentially, such as fixing only the x or y variables, fixing all variables within a class or fixing all classes within a soft distribution constraint. The strategy for fixing the variables is very important; the aim is to find a balance between fixing too many variables, which has a short running time but would produce a small improvement in the quality, and fixing too few variables which allows a wider exploration of the solution space at the expense of increased computational time.

The overall time taken for the first stage of the algorithm ranged from around 5 minutes to 21 hours (5 to 1850 optimization runs), except for the instance “pu-proj-fal19” which required around 260 hours to produce the first feasible solution. The amount of time needed for the first stage depended not only on the size of the problem but also the size of the solution space: small problems with few feasible timetable configurations can be hard to solve. The second stage took between 1 and 240 hours (50 to 5000 runs). Since the second stage is a local improvement metaheuristic, the decision to stop the optimization was partially based on practical considerations and, outside the competition, one could in theory continue these runs to further improve the solution quality.

Algorithm 1 A two-stage algorithm to solve and optimize the timetabling instances

```
{The first stage to obtain a feasible solution}
Solve a minimal MIP containing constraints C-1, C-2, C-5
while Some constraint is violated do
    Read the solution values of the last MIP solved
    Fix randomly some decision variables to their solution values (0 or 1)
    Create a MIP with all constraints C-1 to C-9 satisfied by the current solution
    Add to the MIP all the constraints C-1 to C-9 which are violated, using slack variables
    Optimize the MIP: minimize the sum of the slack variables
end while
return feasible solution
{The second stage to improve the solution quality}
Require: initial feasible solution
while time limit not reached do
    Read the solution values of the last MIP solved
    Fix some decision variables to their solution values (0 or 1)
    Create a MIP containing all the hard constraints
    Add variables and constraints related to student conflicts
    Optimize the MIP: minimize the four cost terms
end while
return solution
```

3 Conclusions and future work

This extended abstract presented a mixed integer programming approach for solving the timetabling problems of the International Timetabling Competition 2019, which produced a ranking of second place in this competition. Although the problem size for a typical timetable was very large to be solved exactly using traditional mixed integer programming tools, several improvements significantly reduced the size to manageable levels. Once a feasible solution was obtained, the use of mixed integer programming for the local search optimization stage proved to be very powerful in improving the quality of the solutions very quickly. A detailed article containing the detailed mathematical formulation, computational implementation, comprehensive results and in-depth analysis is in production and will appear in due course.

References

1. Müller, T., Rudová, H., Müllerová, Z.: University course timetabling and International Timetabling Competition 2019. In: PATAT 2018—Proceedings of the 12th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2018), 5–31 (2018)
2. Burke, E., Eckersley, A., McCollum, B., Petrovic, S., Qu, R.: Hybrid variable neighbourhood approaches to university exam timetabling. European Journal of Operational Research **206**(1), 46–53 (2010)
3. Ergun, Ö., Orlin, J.B., Steele-Feldman, A.: Creating very large-scale neighborhoods out of smaller ones by compounding moves. Journal of Heuristics **12**, 115–140 (2006)
4. Pisinger, D., Ropke, S.: Large neighborhood search. In: M. Gendreau, J.Y. Potvin (eds.) Handbook of Metaheuristics, pp. 399–419. Springer US, Boston, MA (2010)
5. Lindahl, M., Sørensen, M., Stidsen, T.R.: A fix-and-optimize matheuristic for university timetabling. Journal of Heuristics **24**(4), 645–665 (2018)