ITC 2019: Preliminary Results Using the UniTime Solver

Tomáš Müller

Abstract This abstract presents some preliminary results on using the UniTime solver on the International Timetabling Competition 2019 early data sets. The results are compared with the best solutions of the two milestones that are published on the competition website. More results will follow in the PATAT 2020 presentation and in the later version of this paper as the final results of the competitions are made publically available.

Keywords University course timetabling · ITC 2019 · UniTime

1 Introduction

Building on the success of the earlier timetabling competitions, the International Timetabling Competition 2019 (`http://www.itc2019.org`) is aimed to motivate further research on complex university course timetabling problems coming from practice. The competition data sets are based on real-world problems that have been collected using the UniTime application [7]. The individual timetabling problems are quite large with the largest problem having close to 9,000 classes and over 38,000 students. The data for the competition have been collected from 10 institutions around the world and there are a lot of differences between them. For example, some instances have no students (classes are spread in time using hundreds of non-overlapping constraints), some instances are based on student pre-registrations (aka post-enrollment course timetabling) and some instances are based on curricular data. There have been three sets of 10 instances published during the competition: early, middle and late. At the time of writing this abstract, only interim results from the early data sets have been published based on the two milestones.

T. Müller
Purdue University
West Lafayette, Indiana, USA
E-mail: muller@unitime.org

The competition problem combines student sectioning together with standard time and room assignment of individual course events [6]. Classes are organized in a course structure defining the valid combinations of classes a student can take. For example, each student taking a Mathematics course needs to attend a lecture and a lab that is associated with the lecture. The problem also deals with travel times between individual rooms, classes that have different lengths and multiple meetings on a week, classes that are meeting only during certain weeks, and various additional distribution constraints, such as minimizing gaps between classes of an instructor or defining how many class hours an instructor can teach on a day.

UniTime [8] is a comprehensive educational scheduling system that supports developing course and exam timetables, managing changes to these timetables, sharing rooms with other events, and scheduling students to individual classes. It is a distributed system that allows multiple university and departmental schedule managers to coordinate efforts to build and modify a schedule that meets their diverse organizational needs while allowing for minimization of student course conflicts. The software is distributed free under an open-source license and the UniTime project is a part of the Apereo Foundation, a non-profit organization whose mission is to develop and sustain open-source software for higher education.

As the UniTime course timetabling problem is quite complex, with many additional aspects, some simplifications have been made in the competition problem as well as on the problems collected from UniTime. The aim was to reduce the modeling complexity without losing any of the hardness (or computational complexity) of the problems. For example, in UniTime, it is possible for a class to need two or more rooms, or in certain cases, for multiple classes to share a room. Also, some distribution constraints have been removed or reformulated in the competition problem. For example, instead of having a back-to-back constraint, the competition problem requires such classes to be placed in the same room, on the same day, and with the limited time between the first and the last class. This makes for the same outcome when the constraint is satisfied, but the penalization of a partially violated soft constraint is a bit different. More details are discussed in [6].

The paper is organized as follows: in the next chapter, the competition solver is described. There is a short description of the UniTime solver and the code written to make the solver work on the competition problem. Results are presented in the following chapter and conclusions are presented at the end of the paper.

2 The Solver

In this work, the UniTime course timetabling solver is used as it is, even using the default configuration that ships with the UniTime application. New code has been only needed to load the competition problem into the UniTime solver and to save the solution in the competition format. Other than that, some of

the penalizations of violated soft distribution constraints have been changed to follow the competition problem. The code is open-source (under the Apache license) and available in GitHub [3].

The UniTime solver is based on an iterative forward search (IFS) algorithm [7]. This algorithm is similar to local search methods; however, in contrast to classical local search techniques, it operates over feasible, though not necessarily complete, solutions. In these solutions, some classes may be left unassigned. All hard constraints on assigned classes must be satisfied. Such solutions are easier to visualize and more meaningful to human users than complete but infeasible solutions. Because of the iterative character of the algorithm, the solver can also easily start, stop, or continue from any feasible timetable, either complete or incomplete.

The algorithm makes use of Conflict-based Statistics (CBS) [5] to prevent itself from cycling. The IFS algorithm is used until a complete timetable is found. In the next phase, a local optimum is found using a Hill Climbing (HC) algorithm. Once a solution can no longer be improved using this method, the Great Deluge (GD) technique [1] is used. The GD algorithm is altered so that it allows some oscillations of the bound that is imposed on the overall solution value [4].

The solver splits the problem into two sub-problems: student sectioning and class assignment. In the beginning, students are assigned to individual classes following their course demands and course structure. Students with similar courses are kept together as much as possible, using a simple construction heuristics while sectioning one course at a time. This allows for the computation of potential student conflicts between individual classes, that is, the numbers of students assigned to pairs of classes that are overlapping in time or are one after the other in rooms that are too far apart. During the solver run, classes are assigned in times and rooms while the number of student conflicts is minimized, together with the other penalizations on assigned times, rooms, and violated soft distribution constraints. When the class assignment solver is finished, a local-search technique is used to move students between alternative classes or to swap two students between such classes. During the class assignment, student conflicts between two classes that have some alternatives are weighted less (0.2 of the weight defined in the problem) than the conflicts between classes with no alternatives (i.e., conflicts that cannot be removed by re-sectioning).

More details about the UniTime solver, including various improvements that have been done over the years, are presented in [4].

3 Results

The best and the average penalty from 50 independent runs are presented in the following table. The results were computed using Mac Pro (Mid 2012) with two 6-Core Intel Xeon processors running at 3.06 GHz, 64 GB memory, OS X 10.15 and Java 8. The solver uses only one CPU core, and the time

Table 1 Solver results compared with the two milestones.

Instance	M1	M2	UT Best	Average	Time	Room	Dist	Students
agh-fis-spr17	9,259	7,270	**4,687**	5,696.7	236.8	518.4	141.5	421.7
agh-ggis-spr17	98,868	49,901	**47,461**	51,837.9	1,219.0	687.5	1,163.3	5,765.0
bet-fal17	327,048	303,399	**293,925**	295,912.8	223.8	8,827.7	8,942.6	896.4
iku-fal17	74,335	**19,080**	27,867	29,553.1	21,401.8	5,972.8	72.6	0.0
mary-spr17	26,825	**14,927**	15,228	16,034.3	886.0	145.3	2,702.7	60.4
muni-fi-spr16	6,918	**4,112**	4,122	4,336.0	218.5	214.3	3.6	686.0
muni-fsps-spr17	33,760	5,601	**885**	2,002.6	2.5	89.5	67.5	8.4
muni-pdf-spr16c	125,938	85,248	**52,005**	55,905.4	5,364.6	5,143.4	448.5	1,464.3
pu-llr-spr17	34,962	**10,046**	11,387	11,956.9	934.8	1,979.9	52.2	792.2
tg-fal17	8,990	**4,215**	5,795	6,644.0	1,938.6	1,209.9	77.8	0.0

limit was restricted to two hours. UniTime solver cpsolver-1.3.189 was used in the experiment. All the runs were done with the same parameters (using the UniTime's default solver configuration), without any parameter tunning or consideration of a particular instance. The results are compared with the best solutions from the two competition milestones.

Table 1 shows the results from the experiment compared with the best solutions from the ITC 2019 from the two milestones that have already been published. The columns *M1* and *M2* have the two milestones, *UT Best* shows the penalty from the best solution of the 50 independent runs and the Average shows the average result. The best solutions are indicated in **bold**. The last four columns show how the average penalty is split between time, room, distribution penalties and student conflicts.

The solver was able to produce a solution that is better than the two milestones in five cases. Surprisingly, it did the worst on the two instances that have no students (iku-fal17 and tg-fal17; comparing the *M2* results with the *UT Best*), see [2] for more details about the early instances. Out of the five instances where UniTime solver is better than the two milestones, only agh-ggis-spr17 is so close that the average result is worse than the second milestone. In three instances, agh-fis-spr17, bet-fal17, and muni-pdf-spr16c, the UniTime solver has provided a better solution in every single run than the milestone.

4 Conclusion

While the UniTime solver is doing quite well, it has been beaten on half of the early instances already. Better results can be achieved with longer run times and some parameter tuning, which has only been done to a very limited extent. Some additional improvements can be done as well, e.g., students can be re-sectioned continuously during the search, or some of the complexity of the solver (that is not needed for the competition) can be removed.

On the other hand, it is good to see that the competitors are able to produce results that are in par or better than what UniTime would produce out of the box. If this abstract is accepted, a comparison with the final results will be presented at the conference.

References

1. Dueck, G.: New optimization heuristics: The great deluge algorithm and the record-to record travel. Journal of Computational Physics **104**, 86–92 (1993)
2. ITC 2019: International timetabling competition 2019 – Early instances. `https://www.itc2019.org/early-instances`
3. Müller, T.: UniTime ITC 2019 solver source codes. `https://github.com/tomas-muller/cpsolver-itc2019`
4. Müller, T.: University course timetabling: Solver evolution. In: Practice and Theory of Automated Timetabling 2016 Proceedings, pp. 263—-282 (2016)
5. Müller, T., Barták, R., Rudová, H.: Conflict-based statistics. In: EU/ME Workshop on Design and Evaluation of Advanced Hybrid Meta-Heuristics (2004)
6. Müller, T., Rudová, H., Müllerová, Z.: University course timetabling and international timetabling competition 2019. In: Practice and Theory of Automated Timetabling 2018 Proceedings, pp. 5–31 (2018)
7. Rudová, H., Müller, T., Murray, K.: Complex university course timetabling. Journal of Scheduling **14**(2), 187–207 (2011)
8. UniTime: University timetabling – Comprehensive academic scheduling solutions. `https://www.unitime.org`