"Mircea cel Batran" Naval Academy Scientific Bulletin, Volume XIX – 2016 – Issue 1
Published by "Mircea cel Batran" Naval Academy Press, Constanta, Romania // The journal is indexed in: PROQUEST / DOAJ / DRJI / JOURNAL INDEX / I2OR / SCIENCE LIBRARY INDEX / Google Scholar / Crossref / Academic Keys / ROAD Open Access / OAJI / Academic Resources / Scientific Indexing Services / SCIPIO

PRACTICAL ASPECTS ON AUTOMATIC GENERATION OF UNIVERSITY TIMETABLES – A CASE STUDY

Andrei BAUTU[1]
Elena BAUTU[2]
[1]Lecturer, Ph.D., "Mircea cel Batran" Naval Academy, Romania
[2]Lecturer, Ph.D., "Ovidius" University, Romania

Abstract: *The problem of automatic generation of university timetables have been widely discussed in the literature, with many proposed general solutions, from simple heuristics to advanced hybrid algorithms. These algorithms perform well on various test cases, but when they are applied to an instance of the problem specific to an organization, one has to define or adapt the constraints to the particularities of that organization. This adaptation is required for various reasons like algorithm runtime or timetable acceptance from the university staff. In this paper, we present a case study on generating the timetable of the "Mircea cel Bătrân" Naval Academy.*

Keywords: *timetable, genetic algorithm, constraints satisfaction problem*

Introduction

The problem of automatic generation of university timetables is widely discussed in the literature [2], with many proposed general solutions, from simple heuristics [1][8] to advanced algorithms [7][4][3][9][10][11].

The university course timetabling problem is an NP-complete problem that consists in scheduling various teaching activities (courses, laboratories, projects) within a university academic year or semester. The timetable must satisfy a series of constraints related to time, space, and other types of resources (students, teachers, equipment, blackboards, simulators, etc.) used during the specified teaching activities. The teaching activities are scheduled in fixed timeslots, typically spread across one (uniform timetable) or two weeks (odd/even timetable).

Constraints related to teaching activities are restrictions related to the time and space that activities can be planned in. For example, a typical space constraint is related to the capacity of rooms (i.e. rooms need to be large enough seat-wise so that they fit all the students in the class). A typical time constraint is related to the availability of teachers (e.g. some teachers can have teaching hours in the morning).

Problem constraints

When building the timetable of any universities (manually, semi-automatically, or automatically), the staff responsible for this process has to take into account a series of constraints related to teaching staff, students, and resources. Although they may differ in their goal, these constraints fall into two categories: hard constraints and soft constraints. To create a valid timetable (i.e. a timetable that can be used), all the hard constraints must be satisfied (i.e. it is mandatory to fulfill their requirements). On the other hand, soft constraints are not mandatory and some of them can be broken (i.e. not fulfilled). However, the more soft constraints are satisfied, the better the timetable is. Some examples of hard constraints are:

- a teacher teaches only one class at a time
- a room is used by only one class at a time
- a student attends only one class at a time

Some examples of soft constraints are:

- a student should attend maximum X classes per day
- a teacher should teach maximal X classes per day
- students should have at maximum one free timeslot in their schedule per day

Depending on the policies of universities, some constraints that are soft in one university could be hard for others. For example, the decision to give a mandatory lunch break at noon for all students.

FET software

FET (an acronym for Free Evolutionary Timetabling) is a free cross-platform, written in C++ with QT-base user interface, which can be used for scheduling the timetable of schools and universities. It is used or referenced by various research papers [10][11][5][6] because it implements many constraints related to teachers, students groups, rooms, etc. It uses a recursive swapping heuristic to build the timetable automatically, although early versions used genetic algorithms. It also supports manual and semi-automatic scheduling. The authors describe the FET algorithm as a "heuristic placing the activities in turn, starting with the most difficult ones and swaps activities recursively if that is possible to make space for a new activity, or, in extreme cases, backtracks and switches order of evaluation" [12].

DOI: 10.21279/1454-864X-16-I1-005

Automatic timetabling in practice

Automatic timetable software, in general, offers many types of constraints that need to be configured and used to model the policy of the university related to timetables. FET, in particular, offers over 60 types of time constraints and over 25 types of space constraints that apply to students, teachers, and classes.

In this section, we present a case study related to the practical use of FET constraints in automatic timetabling for the "Mircea cel Bătrân" Naval Academy (MBNA). The timetable in MBNA spans on a two weeks period (odd/even timetables), with each week-day having 6 teaching slots (60 timeslots per timetable). The university has 106 teaching rooms across 9 buildings in the campus, which differ in the number of seats, features and equipment, and purpose (laboratories, lecture halls, etc.). The university has a teaching staff of 97 members. They teach classes to 106 cohorts of students, organized in 57 groups (a group has 1-3 cohorts in it, depending on the number of students in the study program), which are organized in 41 study years (i.e. year 1-4 for various study programs). Some classes are taught for individual cohorts (e.g. laboratories, practical projects), some are taught for groups (seminaries), and others are taught for entire study years (course lectures). In total, 1506 teaching activities have to be timetabled in the 60 timeslots.

Some of the hard constraints for the timetable, derived from the university policy, are:

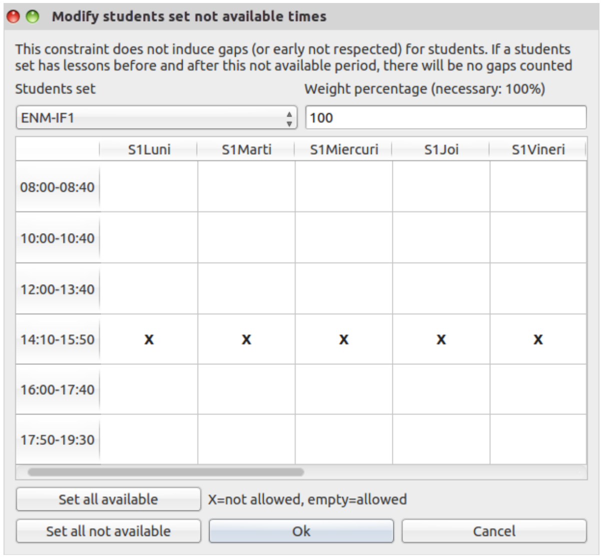
H1. A teacher teaches only one class at a time
H2. A student attends only one class at a time
H3. A room is used by only one class at a time
H4. A room must have enough number of seats to fill all students in the class
H5. Some students cohorts (Navy students) have mandatory lunch break at noon (in the 4th timeslot)
H6. Some students cohorts (e.g. master studies) have classes only in the evening (5th and 6th timeslot)
H7. Some student cohorts have to start classes each day at 8 AM (1st timeslot)
H8. Teachers with management duties must have no classes on Monday mornings (1st timeslot) due to regular management meetings planned at that time
H9. Optional classes must be scheduled at the end of students' day, after mandatory classes (i.e. there must be no mandatory classes after optional classes)
H10. Timetables must have no gaps for students (except of course for free days and inter-day free time)
H11. Course lectures should be scheduled early in the day (1st or 2nd timeslots), except for study programs that have only evening hours (e.g. masters studies).
H12. Students must have at least 2 classes per day (except for free days)
H13. Students must have at maximum 4 classes per day

Some of the timetable soft constraints derived from experience of staff building the timetable manually are:

S1. Related classes (classes for the same discipline) should be scheduled either on different days or in same day one after the other
S2. Teachers should not have more than 3 consecutive classes
S3. Some classes (e.g. lectures, seminaries) should be taught in rooms assigned to the particular study group (e.g. course lectures for the students in the first year of the Maritime navigation study program are taught in room L352).

The H1, H2, H3, and H4 are mandatory hard constraints of FET. They cannot be removed or convert into soft constraints because the resulting timetables might be invalid.

The H5 constraints refer to cohorts of Navy students from different study programs that must have a mandatory break in the 4th timeslot (between 14 and 16 o'clock). There are in total 6 such constraints in our dataset.



This constraint does not induce gaps (or early not respected) for students. If a students set has lessons before and after this not available period, there will be no gaps counted

Students set: ENM-IF1
Weight percentage (necessary: 100%): 100

	S1Luni	S1Marti	S1Miercuri	S1Joi	S1Vineri
08:00-08:40					
10:00-10:40					
12:00-13:40					
14:10-15:50	x	x	x	x	x
16:00-17:40					
17:50-19:30					

X=not allowed, empty=allowed

Figure 1. Example of H5 hard constraint type (mandatory lunch break)

The H6 constraints refer to cohorts (part-time study programs, master study programs, etc.) that have classes in the evening, starting with 5th timeslot (after 16 o'clock). There are in total 19 such constraints in our dataset.

Both H5 and H6 constraints can be implemented in FET using the "Students set not available times" constraint (Fig 1), which is mandatory a hard constraint.

DOI: 10.21279/1454-864X-16-I1-005

FET uses a non-deterministic heuristic algorithm that can produce different timetables on each run. To get some comparable results between different setups, we ran for this study 10 runs for each setup and reported average and standard deviation of runtimes required to produce a valid schedule. Table 1 presents the average and standard deviation of runtime needed to build a correct timetable with various constraints.

Setup (constraints)	**No. of constraints**	**Average runtime**	**St. dev.**
H1-H4	2	4.40	0.52
H1-H5	8	4.40	0.52
H1-H4 and H6	21	4.60	0.52
H1-H6	29	4.60	0.52
H1-H7	51	6.20	0.63
H1-H8	57	7.00	0.82
H1-H9	58	7.20	0.79
H1-H10	59	12.50	1.72
H1-H11	60	6.60	0.52
H1-H12	78	6.10	0.87
H1-H13	79	6.80	1.35
H1-H13, S1	511	11.40	1.71
H1-H13, S1-S2	512	25.70	7.76
H1-H13, S1-S3	590	38.26	9.85

Table 1. Runtimes (in seconds) used for building timetables with different constraints

The H7 constraints refer to cohorts of Navy students from various study programs and full-time students that must start classes in the 1st timeslot of the day (at 8 o'clock). It is implemented in FET as "Students must arrive early" constraint (applied for every day). There are in total 22 such constraints in our dataset. As can be seen from Table 1, H5 and H6 did not increase runtime significantly (compared to the previous configuration, with H1-H4 constraints), but the H7 constraints did increase it by 35% compared to the previous setup. We also attempted to implement this constraint by using constraints on activities (e.g. "Activities have a preferred starting time" in FET), but the runtimes increased significantly because the number of activities is much larger than the number of cohorts, thus needing more computational time to check if constraints are satisfied.

The constraint H8 is similar to constraints H5, but it applies to teachers instead of students and it affects different timeslots. More exactly, the Board of Directors of MBNA has regular meetings every Monday morning at 8 AM (1st timeslot). The teachers that are part of the Board must attend these meetings and therefore cannot attend classes. This constraint is implemented in FET using the "Teacher not available time" constraint. There are in total 6 such constraints in our dataset.

The constraint H9 is used to create timetables in which students do not have to wait idly for mandatory classes due to optional classes. It is implemented in FET using the "Activities end student day" constraint. By using filters provided by FET on the dataset (i.e. based on discipline code) only one constraint instance is used for this rule, but it affects 42 optional classes.

The H10 constraint forces valid timetables to offer students compact schedules (i.e. without gaps) to avoid students the discomfort of waiting between classes or making multiple trips between home and university during the same day. To be noted that this constraint is compatible with H5, i.e. mandatory lunch breaks are not considered as gaps in the timetable. This constrained is implemented in our dataset using one instance of "Students max gaps per week" rule (set up with a maximum of 0 gaps).

The H11 constraint is used to plan lectures in morning hours, to maximize the performance of the teaching process by capitalizing on students' attention. This constraint affects 280 activities, but it is implemented using activities filters in our dataset and only one instance of the "Activities have preferred starting times". An interesting aspect regarding to this constraint is that it reduced the runtime required for finding a valid solution (compared to the other constraints) because it reduced with 66% the size of the search space for valid timeslots for the 280 lecture activities.

The H12 constraint forces valid timetables to plan for students at least two classes per day (except for free days). This is used to reduce the travel overhead to and from university, relative to the number of classes in that day (in combination with H10). This constraint applies to 929 activities, but it is implemented using only 18 FET constraints of type "Students set min hours daily" and additional filters on the activity data set.

The H13 constraint forces valid timetables to have at maximum 4 classes for each student cohort. This is a mandatory restriction to avoid student fatigue. It is implemented in FET using the "Student max hours per day" rule. This rule can also be used for soft constraints, but in our experiments for such case, FET required 10 times more running time to find valid timetables.

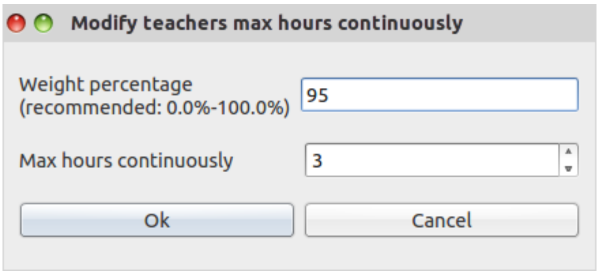
Table 2 presents the average and standard deviation of the conflicts that were caused by broken soft constraints (these values are 0 for experiments using just hard constraints, so they are not included in the table).

DOI: 10.21279/1454-864X-16-I1-005

Setup (constraints)	**No. of soft constraints**	**Average**	**St. dev.**
H1-H13, S1	432	94.43	10.66
H1-H13, S1-S2	433	106.12	14.32
H1-H13, S1-S3	511	120.12	16.25

Table 2. Conflicts caused by soft constraints on the timetables

The S1 constraint is used to allocate a few days between related classes (classes for the same discipline). This extra time will give students time to comprehend the topics discussed in class before moving on to the next topic. However, if for any reason related classes must be scheduled in the same day, then the related classes should be scheduled one after the other to avoid moving students and teachers between rooms. It is implemented using 432 "Minimum days between activities" rules, with a variable number of days (depending on the number of related classes within two weeks) and an acceptance threshold of 95%.


Modify teachers max hours continuously

Weight percentage (recommended: 0.0%-100.0%): 95

Max hours continuously: 3

Figure 2. Example of S2 soft constraint type (maximum continuously teaching classes)

The S2 soft constraint attempts to limit to 3 the number of consecutive classes that a teacher has to attend in any day. Since this is a soft constraint there are occasions when it is broken for some teachers (e.g. teachers with a high load in a given semester). It is implemented using one "Teachers max hours continuously", using a limit of 3 and an acceptance threshold of 95% (see Figure 2).

The S3 soft constraints attempt to schedule various classes in generic rooms (i.e. rooms without specialized equipment or purpose), based on different criteria (related to year and study program of students). They are implemented using filters on the dataset and 78 "Activity tag preferred rooms" rules with 95% acceptance threshold.

In addition to constraints mentioned above, the actual timetabling setup for FET could also consider the personal preferences of teachers (free days, free hours, etc.). Currently, they can only be implemented in FET as a hard constraint. We did not include them in this study as these requirements are not part of the timetabling policy of the university and they are very subjective (i.e. vary from semester to semester).

CONCLUSIONS

In this paper, we presented a case study on using open source software to implement the timetabling policy of the university, with the goal of automatically generating valid timetables for teaching activities. The results presented include runtime statistics and implementation details that can be used in other educational institutions to obtain similar results. As we noted in previous sections, some of the timetabling constraints can be implemented in different ways, leading to different outcomes (from minute to days of runtime, or from equally good to impossible timetables).

BIBLIOGRAPHY

[1] Anmar Abuhamdah, et al. "Population based Local Search for university course timetabling problems." Applied intelligence 40.1 (2014): 44-53.
[2] Hamed Babaei, Jaber Karimpour, and Amin Hadidi. "A survey of approaches for university course timetabling problem." Computers & Industrial Engineering 86 (2015): 43-59.
[3] Rakesh P. Badoni, D. K. Gupta, and Pallavi Mishra. "A new hybrid algorithm for university course timetabling problem using events based on groupings of students." Computers & Industrial Engineering 78 (2014): 12-25.
[4] Ruey-Maw Chen, and Hsiao-Fang Shih. "Solving university course timetabling problems using constriction particle swarm optimization with local search." Algorithms 6.2 (2013): 227-244.
[5] César Covantes, and R. Rodr. "The Design of Multi-agent System for the Solution of School Timetabling Problem." 2015 Fourteenth Mexican International Conference on Artificial Intelligence (MICAI). IEEE, 2015
[6] K.L. Ertürk, G. Sengül, and M. Rehan. "How to Use Cobit Applications in Educational Institutes." International Journal of Management and Sustainability 3.2 (2014): 42.
[7] Cheng Weng Fong, et al. "A new hybrid imperialist swarm-based optimization algorithm for university timetabling problems." Information Sciences 283 (2014): 1-21.
[8] Joe Henry Obit, et al. "An Evolutionary Non-Linear Great Deluge Approach for Solving Course imetabling Problems." IJCSI International Journal of Computer Science Issues 9.4 (2012): 1-13.
[9] Khalid Shaker, et al. "Hybridizing meta-heuristics approaches for solving university course timetabling problems." Rough sets and knowledge technology. Springer Berlin Heidelberg, 2013. 374-384.

DOI: 10.21279/1454-864X-16-I1-005

[10] Rafal Tkaczyk, Maria Ganzha, and Marcin Paprzycki. "AgentPlanner-agent-based timetabling system-preliminary design and evaluation." System Theory, Control and Computing (ICSTCC), 2013 17th International Conference. IEEE, 2013.
[11] Rafal Tkaczyk, Maria Ganzha, and Marcin Paprzycki. "AgentPlanner-agent-based timetabling system.", Informatica 40.1 (2016).
[12] ***, FET software home page http://lalescu.ro/liviu/fet/

DOI: 10.21279/1454-864X-16-I1-005