Constraint Priorities - a Way of Getting an Optimal Timetable Fully Automatically. Three Steps Into the Modern Timetable Scheduler

Wieslaw Dudek
Wieslaw Dudek Timetables, Krakow, Poland
Wieslaw.Dudek@gmail.com

Abstract

The presentation goes through all the system features which make the new timetabling technology unique and efficient, presenting constraint priorities as the most important innovations in the scheduling process. The system of priorities exists in the main three fundamental modules: inconsistency detection system, optimization module and solver. The complete constraint system is also presented as an important part of useful timetabling software.
Moreover, the presentation covers other features which can also make the solution useful for other non-school problems. The described solution is available at www.school-timetable.eu.

Introduction

The process of building a timetable is very complex and arduous as it requires many difficult constraints to be reconciled at once. This is too great a challenge for people even in cases of small bundles of requirements but can be simplified by using modern software systems.

In this paper the modern system means software which can take care of the whole scheduling process without bothering its users too much and without asking any questions about how to arrange some classes which apparently had been too difficult for the system to set. All the questions should be constructive ones.

The www.school-timetable.eu site is such a new, modern solution to the problem. The most innovative feature is prioritizing the constraints. It is so natural for people working on a timetable to abandon some unimportant requirements due to lack of time or skills. Time and skills count in virtual reality as well.

Constraint priorities are taken into account in the main three fundamental modules in the system i.e. inconsistency detection system, optimization module and solver.

Every scheduler needs a complete set of requirements, which can be defined by the user, to achieve the goal of getting a timetable in a fully automatic way, simply by clicking on „Generate" button. Without a complete set, the received solution would be imperfect and would require user interaction and analysis; in some cases the whole timetable would need to be rebuilt making the solution useless.

If users are given a possibility to enter all of their requirements into the timetable, they most likely enter all of them and thus schedulers need good inconsistency and optimization systems based on priorities.

Inconsistency Detecting System

The inconsistency detecting system filters the requirements and removes the ones with less priority which are in conflict with the important ones. The priorities can be assigned to constraints based on their category i.e. min, max quantity, gap mode etc. and the resource they concern e.g. a teacher, a classroom.

Assuming the priorities were assigned to constraints correctly, the system will erase only the ones that must be removed i.e. the ones with lower priorities.

Solver

As soon as the inconsistency detecting system removes inconsistencies, the solving process begins. This process needs to be efficient and it is crucial for schedulers to consider it effective. The www.school-timetable.eu system has got the most powerful solver. The solver was tested on data coming from 2008-2010. The 4.0 version proves its effectiveness by arranging *all* of the classes with good execution time, provided the user turned on the correction process or has not banned any of the constraints from being removed during the optimization process. The algorithm managed to solve a timetable with 13 sites, 1300 courses, 12000 subjects, 30000 students in it, which means it can be used for timetabling a university schedule with shared

classrooms, teachers etc.

The solver takes into consideration each requirement which needs to be defined for school timetables and can be easily extended with new ones. At present the solver copes with all of the constraints used in school planning that were necessary during the last few years of the system development, even at times when full timetables needed to be built without enough information provided e.g. missing classes, unknown language levels.

The quest for completeness produced flexibility of the system and now it is able to handle other types of timetables e.g. work schedules. It is worth mentioning such constraints, which are not present or very rare in other timetabling systems, like different working times for sites, inter-moving modes and times, classes order, first/last lesson in a day, packages of courses for individual students, resources other than classrooms, combined & correlated classes, shift work, space between classes etc.

Optimization Module

Often to get a complete solution some systems need their users to manually abandon some difficult constraints and decide which ones need to be removed first. Our system is much more „aware" of timetabling possibilities and reality than its users; everything it needs from them is the importance of constraints in a form of their priority. There is no reason to ask the users to help solve some sub-problems because the computer has much more chance to try them out quickly.

The optimization process is a way of getting a complete solution. The last step in it is the correction process, which is the last resort to receive a full solution if the user had not allowed to remove some of the constraints (called fixed) because of their high importance. The correction process works in two runs. The first run will be called here „the ambitious run" and results in a fully or partially built timetable with all important (fixed) constraints in their place. The second run, called „the good enough run," arranges the rest of the classes which were not able to be arranged in the previous „ambitious run." However, this time without constraint fixing so it will be possible to remove them if necessary and place all the remaining classes. This is modeled after people who need to abandon their ambitious plans before they fail; doing so at the very last moment they are able to get much further, had they taken a shortcut right at the beginning.

The timetable correction is fully automatic but can also be used manually if the users want to. It also lets its users fix more constraints than it is really needed if they are afraid they could be removed by the system too early.

System Architecture

The ideas mentioned in the preceding paragraphs were foundations of the www.school-timetable.eu system architecture depicted on Figure 1.

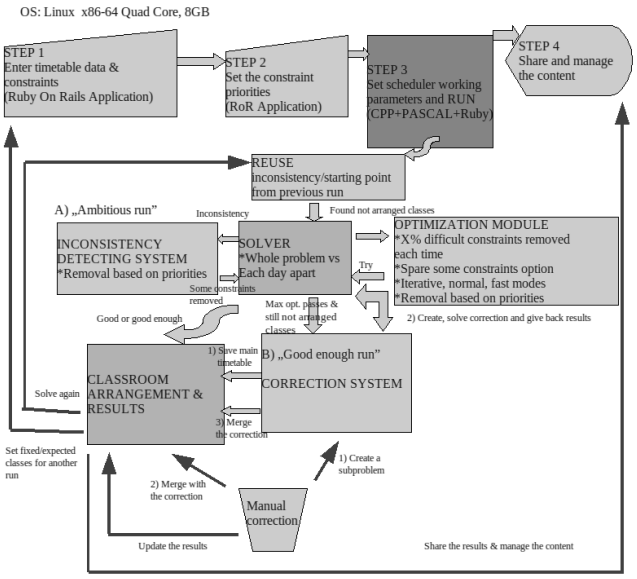

Figure 1: Overview of the www.school-timetable.eu system architecture

As one can see from the process flow perspective showed on Figure 1, the system fully automatically applies a wide range of techniques in order to receive a complete solution. There is no need for difficult questions like "How to arrange some classes?" because the system can decide by itself which difficult constraint it should remove on the basis of constraint priorities. However, a user is allowed to change the priorities dynamically by creating corrections (see „Manual correction>Create a subproblem") or even arbitrarily impose a given distribution of classes manually (see „Manual correction>Update the results").

Besides, it is possible to define an expected or fixed distribution of classes at the very beginning if one is aware of all of the requirements at this stage. Taking a closer look into the remaining steps of the process we can distinguish the following ones: entering data /constraints defining/, assigning priorities to constraints and sharing and managing the final timetable.

Constraints Defining

The step of entering data needs to be sufficient and effective. Sufficiency means a complete set of constraints which can be reflected in the system (see „Solver" chapter) while effectiveness signifies a simplicity of applying constraints. For many scheduling systems defining groups

of students seems to be the most difficult thing to do.

Creating groups is related to many reasons; one of them concerns student body sizes - some of them are too big to be taught, other ones suffer from the shortage of teachers. Another reason for dividing students into groups can be a need to combine groups on student body, year or school level. At the end students can select an individual education mode or some optional courses. The system has to be on alert and ensure that some common constraints for such groups will be met e.g. to avoid overlaping of courses selected by the same student, impose no gaps for students in certain kinds of schools or make sure that some defined constraints for the groups will be met.

In www.school-timetable.eu scheduler system the way of defining groups depends on the initial data we have. If we already know the way of dividing students into groups, the whole process can be simplified by using packages of courses (see Figure 2).

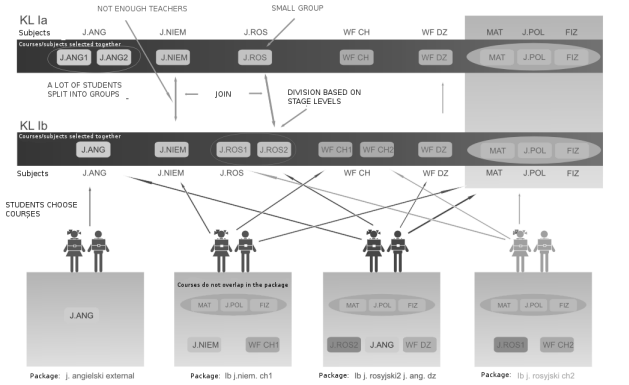

Figure 2: Defining the packages of courses

Sometimes it is impossible to say how students are going to be divided, nonetheless, we need to secure a certain selection of groups avoiding overlaping of courses; later on appropriate courses will be selected according to skills or preferences. To achieve this goal we cannot use packages of courses but need a more flexible solution - correlating subjects designed by a maximum number of simultaneous lessons from a group of subjects or a subject combining feature.

Setting a minimum number of simultaneous lessons can be a way of correlating some classes of subjects; however, setting a max number of simultaneous lessons to one could be a way of separating subjects in time. On the other hand, the goal of subject combining is to correlate lessons, share a classroom by some of the subjects, define a common lesson for several classes, define elective line i.e. a block of many classes of many subjects where each student may choose one subject from that line.

Priorities Defining

The second step of the flow in Figure 1 is to assign priorities to the previously entered constraints. To simplify the assignment process there are some predefined priorities for existing constraints which do not have to be redefined in many cases. However, if necessary, they can be changed e.g. a user can demand no gaps for a few teachers as a priority request.

In the example below (Figure 3) John Smith and Ann Brown's teacher gaps & availabilities are set to be more important than the same constraint types for other teachers. Also John Smith's gaps & availabilities are more important that the same ones for Ann Brown because "Elements before constraints" attribute is set to "Yes". With the "No" value the order would be different: John Smith's gaps > Ann Brown's gap > John Smith's availability > Ann Brown's availability.

PRIORITY	RELATED TO	REQUIREMENTS	ELEMENTS	ELEMENTS BEFORE CONSTRAINTS
1	Subjects	Number of lessons per week	All subjects	No
2	Classes	No gaps	All classes	Yes
3	Subjects	Classrooms arrangement	English; Chemistry	No
4	Classes	Availability; Starting time of lessons	All classes	No
5	Teachers	Gaps constraints; Availability	John Smith; Ann Brown	Yes
6	Teachers	Availability	All teachers	No
7	Subjects	All	All subjects	No
8	Classes	All	All classes	No
9	Teachers	All	All teachers	No
10	Groups of subjects	All	All Groups	No
11	Students	All	All students	No
12	Resources	All	All resources	No

Figure 3: Example of constraint priorities defining

Sharing and Managing the Final Timetable

Ultimately, after a timetable is created, it can be easily accessed online by students, classes, teachers & classrooms supervisors. If required, registering for courses can be turned on and even more preferences can be taken into account during the scheduling process. Another option for schools is the possibility to manage teacher substitutions online. The change will be immediately communicated to the end users /students, teachers etc./.

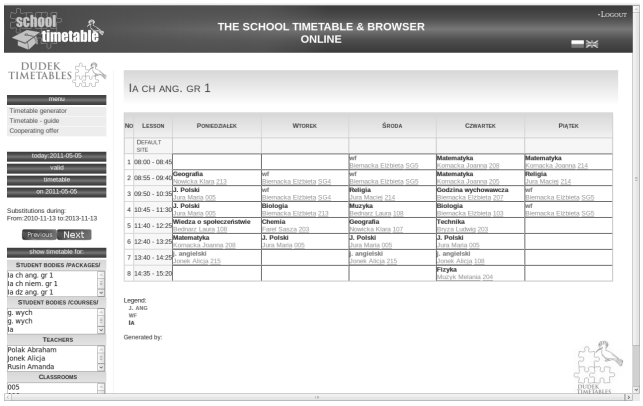

Figure 4: Sharing the timetable online

Development Perspectives

The author's desire is to build a modern and user-friendly tool for scheduling purposes both for schools and other institutions working with timetables; the system should be capable of creating optimal timetables and should allow its users to access them online. The release of the Android version of the timetable browser is planned in 2011. The new interfaces for institutions other than schools should be soon put into place.

Although the algorithm is in its final state of development there are still opportunities for improvement. Generating a timetable for a university with all of its departments is available, though the whole process takes a few hours. It is possible to increase its speed by using concurrent threads and splitting techniques.

Another area for development is an classrooms arrangement system which could be capable of operating on groups of classrooms instead of individual ones. It is worth mentioning the another area for improvement such as integration with other systems. Although the system is ready to operate on the open JSON RPC standard and it is also fast enough to be used separately, the integration with other existing systems can be convenient for the end users. It is possible to develop some pieces of administrative software to use with the timetable generator or to integrate it closely with other systems which perform their tasks well.

History

The idea of the scheduler in its current state has been maturing for 16 years since 1995 when it was first taken into consideration. The year 2008 turned out to be critical for the project since the main questions were answered and a new online system was built.

However since its first release in 2008 many parts of the generator have been improved. Here are some of the most important dates in the project's development. Most of the following tasks were executed by one person only - the author:

- 1995-2008 - idea development,
- IV 2008 - the beginning of the project and its first release - v1.0,
- V-VI 2008 - improvements in inconsistency system,
- VI 2008 - gap constraints revised,
- VII 2008 - classroom management system (preferred classrooms, arrangement modes, fixed classrooms) & no 1 or 2-hour-long working days,
- IX 2008 - student groups, combined subjects, more improvements in inconsistency system,
- II 2009 - v2.0 with new graphical design and lots of improvements: sharing & managing timetables online (substitutions); priorities used in inconsistency system; new important constraints introduced: group of days, order of classes, sites with independent set of times of lessons, blocked subjects, students & courses,
- VII 2009 - optimisation system using priorities,
- XI 2009 - v3.0 with usability improvements, packages of courses, registration for courses before or after scheduling, correlation or separation of subjects within subject groups,
- XII 2009 - optimisation system improved,
- III 2010 - v4.0 performance & efficiency improved; JSON-RPC & XSD,
- VIII 2010 - successful execution of large university timetables with up to 13 sites and 1300 courses with about 10 subjects each (12000 subjects were defined),
- XII 2010 - new usage scenarios and application - work scheduling; data entering speed-up and simplicity - mass change & automatic package of courses generation based on courses properties,
- I 2011 - bilingual version and foundation for multilingual version; exports & imports of data,
- II 2011 - timetable correction by days and further speed-up of generation process,
- III 2011 - timetable correction by sub-problem /solving sub-problem in independent way as an external timetable and merge results afterwards/, automatic correction if not arranged lessons; manual and arbitrary corrections.

References

Demo samples: http://www.school-timetable.eu/access/demo

A service guide: http://www.school-timetable.eu/guide/guide

A service tutorial: http://www.school-timetable.eu/access/tutorial

More info: http://www.school-timetable.eu