System for solving timetabling problems

Wojciech Legierski *
Institute of Automatic Control, Silesian University of Technology, Akademicka 16, 44-100 Gliwice, Poland
e-mail: wlegierski@ia.polsl.gliwice.pl

Paweł Parys
Faculty of Mathematics, Informatics, and Mechanics, Warsaw University, ul. Banacha 2, 02-097 Warsaw, Poland
e-mail: pp209216@zodiac.mimuw.edu.pl

Abstract

Timetabling construction is a hard computationally problem. But difficulty lies not only in NP-completeness of that problem but also in complicated interaction between different users. This paper present System of Automated Timetabling (SAT) as a proposition for a global look at timetabling. It describes four tools for managing timetable. For solving problem Constraint Programming (CP) techniques was used. CP provides both declarativity and flexibility. It is easy to add new constraint and to model numerous specific constraints that are often encountered in such problems. Additional local search was incorporated into CP to allow effectively optimising timetable.

Keywords: timetabling, constraint programming, search strategy, user interface

1. Introduction

Timetabling is usual problem of each academic community and it was very often tried to solve by researcher. Although a lot of scientific and commercial work was made in this area, in many educational institutions timetable is still scheduled manually and often if a computer is used, it is need only for data presentation or checking constraint validation. Many functions that can be made automatically by computer are neglected. The reason of this situation is:

- different problem formulation in different universities/schools,
- lack of proper programs that can adapt for special requirements of educational institutions,
- lack of user interfaces for providing interaction between different users.

This paper presents interaction between different information technologies, what gives ability to develop functional system for solving, managing and presenting timetable.

There are a lot of publications describing work on algorithms and heuristics for timetable problems. There are methods using tabu search, simulated annealing, genetic algorithms, ant colony optimisation and constraint programming. Difficulty in comparing them and finding the best one lies in the first mentioned reason – a huge number of problems. Work concerning user interface is almost unseen in research community – some papers only mention about it. From the other side commercial applications for timetabling have a rich user interface, but they often do not bother with automating solving problems and often their data standard is not general enough. This paper presents System of Automated Timetabling (SAT) that try to look at timetabling process as whole. Automated solver is presented briefly. It uses Constraint Programming techniques with additional local search that performs much better than standard branch and bound. A special technique was used for implementing soft constraint called value assessment. Detailed description of used search method can be found in author paper [6]. The main contribution of this paper is proposition of environment for multi-user timetable process.

* The authors research has been supported by the Foundation for Polish Science

2. Users

Timetable programs should not be treated as spreadsheets with additional functionality, which is adapted for a one user, who, after schedule is made, prints it for students. Automated timetabling should be seen rather as a system, where different user can interact between each other. The following users take part in timetabling process.

2.1. Students

Time spending for acquisition of information connected with timetable is very often neglected. Spolsky in [10] mentions a rule that is binding in user interface developing: “a seconds are hours” explaining that if our bad develop interface forces a user to waste a second for finding some features, it sums to hours for all users. If university have thousands of students, it can be imagine how much time could be save when students are able to easy collect and insert information. First of all students should have available to see in easy way all kinds of timetable (their own, classroom timetable for checking if it is free for additional events and conductors timetable for checking availability of their teachers). They should be not only tidy presented but also well prepare for printing them.

In some institutions students’ preferences are taken into account and all of them should be easy inserted by students.

2.2. Conductors

They have always some requirements and preferences that are connected with time and order of their courses. A lot of notes on small scraps, collected by timetable manager do not ensure taken them into account. Possibility of adding additional events (e.g. consultation with student, courses in other schools) to their timetable would be willingly welcome.

2.3. Timetable managers

It must not assume that only one person is responsible for scheduling whole timetable. When a big university is taken into consideration, timetable managers from each department often use the common resources and conductors from one department teach in other departments. Timetable managers should work on the common database with proper privileges for their resources and

have a system of exchanging demands. Scheduling, which would be provided, should be:

- manually with drag'n'drop properties, constraint checking and proposal of proper timeslots for courses,
- automatic with possibility to define a part of courses that are schedule and base only on available for user resources

Automatic timetable should not use only computation power of timetable manager computer, but it should be made on a server with good computation performance.

2.4. Administrator

There is need for a person who describes structure of the timetable, creates users and their privileges. Most often there will be one of the timetable managers. This person can use the same program like timetable manager with additional functionality.

3. System of Automated Timetabling SAT

An idea of a system that connect all users assume:

- availability from different platforms for students and conductors;
- easiness in building interface;
- effectiveness first of all for timetable managers;
- common database.

One tool or only one language is not enough to fulfil all these requirements. Therefore proposed SAT is developed by adding the use of several tools:

1. Database:

MySQL as database gathering information not only needed to solve timetable but also users description and information connected with structure of timetable,

2. Web application:

PHP, JavaScript and HTML for presenting data on the Web pages and giving possibility to insert students and conductors preferences,

3. Timetable Composer:

VC++ for writing program for timetable managers that connects with database through ODBC

4. Solver:

Mozart/OZ for an engine solving automatically timetable problems

Relation between these parts and their data format is depicted on Figure1. Each of these parts can by on different computer. Of course Internet application and database needs continuously working server and for a better performance should be installed on the same computer. The idea of the separation solver and the Timetable Composer lies on requirement that timetable manager does not requires a high performance computer and a computation for an automated timetabling can be made remotely on computer of higher quality (e.g. on the same server where is a database located and an internet application). One of the requirement of the selected tools was that the portability should not be depended on server system. MySQL, PHP and Mozart/OZ can work both on Unix/Linux and Windows platform. One exception was made with Timetable Composer, but it can be mainly run on a client computer and do not need to be adapted to system of existing server.

3.1. Database

Many formal rules or procedures have been proposed for timetabling descriptions. Some of them are dedicated to specific problem.

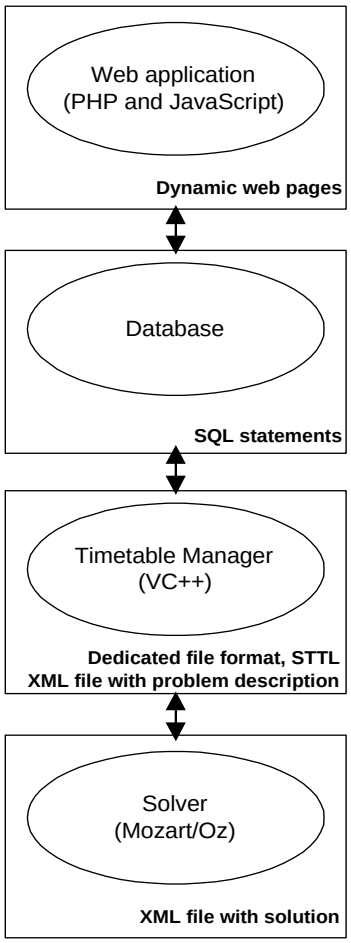

Fig. 1. Diagram of four parts of SAT and their dependencies and output data format

General timetable problem description has been made recently by Kingston [5], Reis and Olivier [8] and De Causmaecker et al. [3]. Kingston introduces an STTL language for specifying and evaluating timetabling problems. It base on TTL specified in Burke et al. [2]. The same specification uses Reis and Olivier [8] and they propose simpler and verbose language for describing timetable problems. Their requirements were an interchange form between a computer specialist and a school administrator. De Causmaecker et al. [3] develop Kingston [5] approach and introduce it into XML standard and describe as ontology.

The requirements that were mention in these works concerning standard data format were following:

- ability to express all kinds of timetabling problems;
- ability to express these problems in full detail;
- easiness to translate to and from.

All these descriptions assume text format. Although it is useful for creating benchmarks, avoid confusion and allow comparing heuristics between researches, it has some drawbacks for real-life multi-user application. Presented system promotes opinion that relational databases can fulfil all requirements mention above and adds user management facilitates, data acquisition by the SQL queries and provides compactness of data. Of course all databases can be described as text format with SQL statements. Small additional programs can provide translation between the mentioned standards and a database.

3.2. Web application

HTML seems to be an obvious solution for presenting data on the Internet, but it provides only static pages what is not sufficient for described system. JavaScript improves user interface and gives capability to create dynamic pages. As a client-side extensions it allows an application to place elements on an HTML form and respond to user events such as mouse clicks, form input, and page navigation.

Server-side Web-scripting languages provide developers with the capability to quickly and efficiently build Web applications with database communication. They became in last eight year a standard of dynamic pages. PHP as one of the most popular scripting language in the world was chosen for developing SAT. Its main advantages are facilities for database communication.

People are used to Web services which provides an arranged news and information, search engines for sites in whole Internet, email application etc. Proposed system try to be similar to those kinds of services in it functionality and generality. Most timetable application gives a possibility to save schedules for particular groups, teacher and class as HTML code, but it not assumes an interaction with its users. Timetabling involves often a lot of users to create it. Student and conductors should have ability to introduce their requirements and preferences into a database. It should manage all these users with passwords and usernames in order to protect the data against changing it by improper users. On Figure 2. is depicted an example of the solution of above requirements. On the left side is a navigation tree with particular resources and in main area is a table with weekdays and their hours. Each conductor can mark his/her preferences by proper colour.

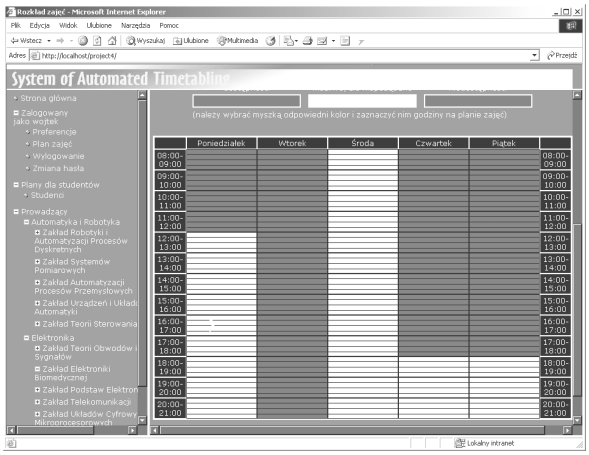

Fig. 2. Sample screenshot of Web application

There is normal situation that final schedule is printed. Not only school administrator makes it, but every student wants his schedule above his bed. Printing from HTML pages often is not looking neat and tidy. Special module that creates Latex files and next generates PDF was implemented. It adapt to changing data in database.

3.3. Timetable Composer

It is hard to develop a fully functionality in Internet technologies. Therefore system for automated timetabling use VC++ for Timetable Composer, program for timetable manager and administrator. The idea of the program is to simplify manual scheduling and provides as many information as possible. Therefore well-known rule drag'n'drop is implemented, layout is based on tree navigation and different views, for each event possible timeslots are showed etc. Example screenshot of Timetable Composer is depicted on Figure 3. Saving data can be made in two ways. First method saves data locally in dedicated file format, the second one use ODBC to save data in remote database. It takes into account privileges of users and do not allow change data for which user has not rights.

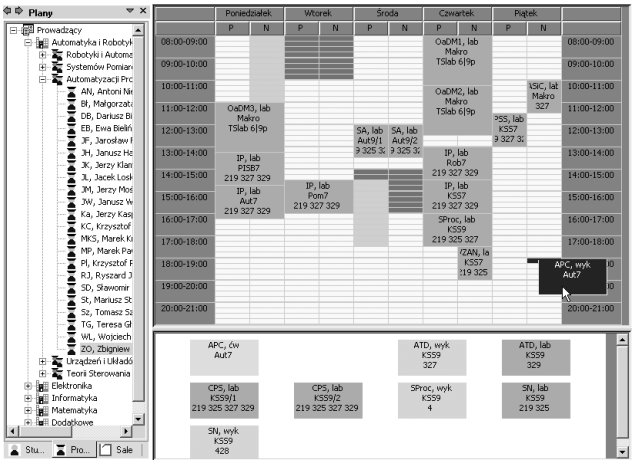

Fig. 3. Timetable Composer

4. Solver

Constraint Programming techniques have been developed since about 1990 and become one of the branch of Artificial Intelligence. Because they base on backtracking search, at the beginning they have been developed in Prolog, where backtracking and declarativity had been already implemented. In this way Constraint Logic Programming (CLP) was created as an addition to Logic Programming (LP). The languages from this area, which are still popular, are CHIP, Sicstus, Eclips to name a few. Then CP leaves a Prolog and comes into two branches – one of them is C/C++ libraries (e.g. ILOG) and the second is multiparadigm languages (e.g. Mozart/OZ). All of these languages have two common features – constraint propagation and distribution (labelling) connected with search. Mozart/OZ [7][10] was chosen to develop SAT because of it unique features that allow to create special distribution strategy and search method.

4.1. Distribution strategy

The main requirements for a good distribution strategy are:

- reduction of backtracks (decrease the frequency of constraint violation),
- searching for a good solution right away (fulfil as many soft constraints as possible).

The first requirement fulfils well-known first-fail strategy, which choose variable with smallest domain. Decision about variable to instantiate become harder if timetable problem with room allocation is considered. Then we need rather finding room and timeslot for course (two variables together) than instantiating each variable separately. In that case choosing proper course is consider and decision is made due several factors. These factors depend on hardness of course schedule (size of start time domain, room domain, load of teacher and fails number).

4.2. Soft constraining

Assessment for values in domain was introduced in solving timetable by Abdennadher and Marte [1]. They represent a domain as a list of value-assessment pairs. For example, X :: [3, 4, 5] can be [(3, 0), (4, 1), (5, 8)]. Next they use Constraint Handling Rules (CHR) for solving Weighted CSP. The idea of value assessment for variables related to start times was applied for solving timetable problem as well as the Weighted CSP framework. The assessment of the value corresponds to the fulfilment of the soft constraint. 0 means full consistency with all soft constraints. If a value does not fulfil soft constraints its assessment is increase corresponding to the weight of the constraint. This framework allows taking into consideration many soft constraints. The assessing of values was made only for start time domain of courses. Kind of soft constraint force the stage of introducing it:

- once at the beginning of the search (e.g. late afternoon hours are not preferred)
- during the search, if soft constraint depends on already schedule courses (e.g. whether the value corresponding to the start time does not make gaps between already scheduled courses)

Additional very useful techniques are removing from domain value with high assessment. This idea is similar to Restricted Candidate List [4], which retain only the good branches in search and discard the bad ones. The drawback of this technique is rejecting possible solution, but for large problems that need also optimisation it is rather rejecting bad solution. Additionally removing value from domain leads to propagation that accelerates finding feasible and good solution.

4.3. Constraining during distribution

Introducing constraints during search such that the different sets of constraints are active in different branches of the search tree is not usually preferable, because this approach provides weak constraint propagation. From the other side in some cases global constraints can be introduced very hard and computational effort of checking them can be high. This case appears when room allocation is performed simultaneously with search for start times. Introducing constraint “one course in a room” is very hard, when most of rooms have special features different from other rooms and most of courses have more than one timeslot duration.

It was resolved by introducing this constraint after instantiation of room and start time of specific course and refers to all not labelled courses. It considers the corresponding room at corresponding time as tentatively off-limits for all other courses. This approach gave much better results than global constraining.

4.4. Improving solution by local search

Improving solution by standard branch and bound (BAB) did not give good results because distribution is design to find good solution right away and BAB is systematic method that make a lot of redundant work. Local search have been proved to obtain very good results in large combinatorial optimisation problems. White and Zhang [11] made a successful approach to combine local search with constraint satisfaction for timetabling problem. They determined an initial solution using constraint logic programming and then optimised it using tabu search. Describing approach tries not to resign from constraint propagation and incorporates local search into constraint programming. The used technique is similar to Large Neighbourhood Search (LNS) proposed by Shaw [9]. After finding solution course, that causes the highest cost, is relaxed with courses depending on it. All other courses have the same start time. Search is made only on relaxed courses and is treated as neighbourhood move. If it produces better solution, it is kept. The tabu list is added not to relax always the same courses.

5. Conclusion

The presented system has been prepared to introduce it in Silesian Technical University. Solver working on the real university institute problem (223 courses, 23 students groups, 54 teachers and 42 rooms and 70 timeslots) was tested on a Pentium III/850 MHz, 256 MB station. The first solution was found after 45s, and 20% improvement of cost function was achieved after next 20s using the presented idea of the local search, whereas BAB did not give any improvements.

Scheduling timetable automatically can save time in comparison with manually timetabling. But experience shows that much more time can be saved if only managing and acquisition of data will be improved. Therefore emphasis is put on developing multi-user system that uses the newest computer science technologies.

References

[1] Abdennadher S., Marte M.: *University course timetabling using constraint handling rules*, Journal of Applied Artificial Intelligence, 14(4), 2000, pp. 311–326.

[2] Burke E.K., Pepper P.A. and Kingston J.H.: *A Standart Data Format for Timetabling Instances*, Practice and Theory of Automated Timetabling, Springer-Verlag 1408, 1998, pp.309-321.

[3] De Causmaecker P., Demeester P., Lu Y., Vanden Berghe G.: *Using Web Standards for Timetabling*, Proceedings of the Fourth International Conference on Practice and Theory of Automated Timetabling, Gent 2002, pp.238-257

[4] Foccaci F., Laburthe F., Lodi A.: *Local Search and Constraint Programming, Handbook on Metaheuristics*, F. Glovert and G. Kochenberger (Eds.) (2003)

[5] Kingston J.H.: *Modelling Timetabling Problems with STTL*, Practice and Theory of Automated Timetabling III, Springer-Verlag 2079, 2001, pp.309-321

[6] Legierski W.: *Search Strategy for Constraint-Based Class-Teacher Timetabling*, Practice and Theory of Automated Timetabling IV, Springer-Verlag 2740, 2003, pp.247-262

[7] Mozart Consortium, *The Mozart Programming System*. Documentation and system available via WWW from http://www.mozart-oz.org, 1999.

[8] Reise L.P., Oliveira E.: *A Language for Specifying Complete Timetabling Problem*, Practice and Theory of Automated Timetabling III, (selected papers from Proceedings of the Third International Conference on Practice and Theory of Automated Timetabling, Konstanz 2000), Springer-Verlag, vol. 2079, 2001, pp.322-341

[9] Shaw P.: *Using constraint programming and local search methods to solve vehicle routing problems*, Principles and Practice of Constraint Programming - CP’98, LNCS 1520, Springer-Verlag, (1998) 417-431

[10] Spolsky J.: *User Interface Design for Programmers*, APress, 2001

[11] White G.M., Zhang J.: *Generating Complete University Timetables by Combining Tabu Search and Constraint Logic*, Practice and Theory of Automated Timetabling II, Springer-Verlag, Lecture Notes in Computer Science, vol. 1408, (1998) 187-198