MISTA 2009

Interactive Course Timetabling

Tomáš Müller · Hana Rudová · Keith Murray

1 Introduction

Timetabling problems are often solved based upon what is assumed to be complete information about the problem. However, users may later desire modifications to the generated timetable to adapt to changing requirements or increase their personal satisfaction with the solution [1,5,3]. Methods that allow corrections to an existing timetable while leaving most of the solution intact are necessary. Furthermore, the ability to make interactive changes to a timetable introduces a desirable approach by allowing users to readily react to a limited set of altered requirements by choosing among appropriate options. The aim of this paper is to introduce such methods of interactive timetabling as they have been applied at Purdue University. The concept of using interactive timetabling in this problem has been described previously [4], however, this work introduces a new formal view of this approach.

2 Applied Approach

Interactive changes to a timetable are easily understood by users as a sequence of changes to individual classes, Figure 1 illustrates the information available to a user considering changes to a class of interest (later denoted v_{bb}). The user may explore different options, consider various types of changes with the class, commit selected choices, or discard all changes considered. *Selected Assignments* describe changes already made to the timetable during the current interaction (denoted μ). *Conflicting Assignments* inform the user of any conflicts created in the timetable (denoted γ) as a result of the selected assignments. *Suggestions* (denoted Ω) are proposed options the user may choose from to make the timetable feasible.

T. Müller · K. Murray
Space Management and Academic Scheduling, Purdue University
400 Centennial Mall Drive, West Lafayette, IN 47907-2016, USA
E-mail: muller@unitime.org, kmurray@purdue.edu

H. Rudová
Faculty of Informatics, Masaryk University
Botanická 68a, Brno 602 00, Czech Republic
E-mail: hanka@fi.muni.cz

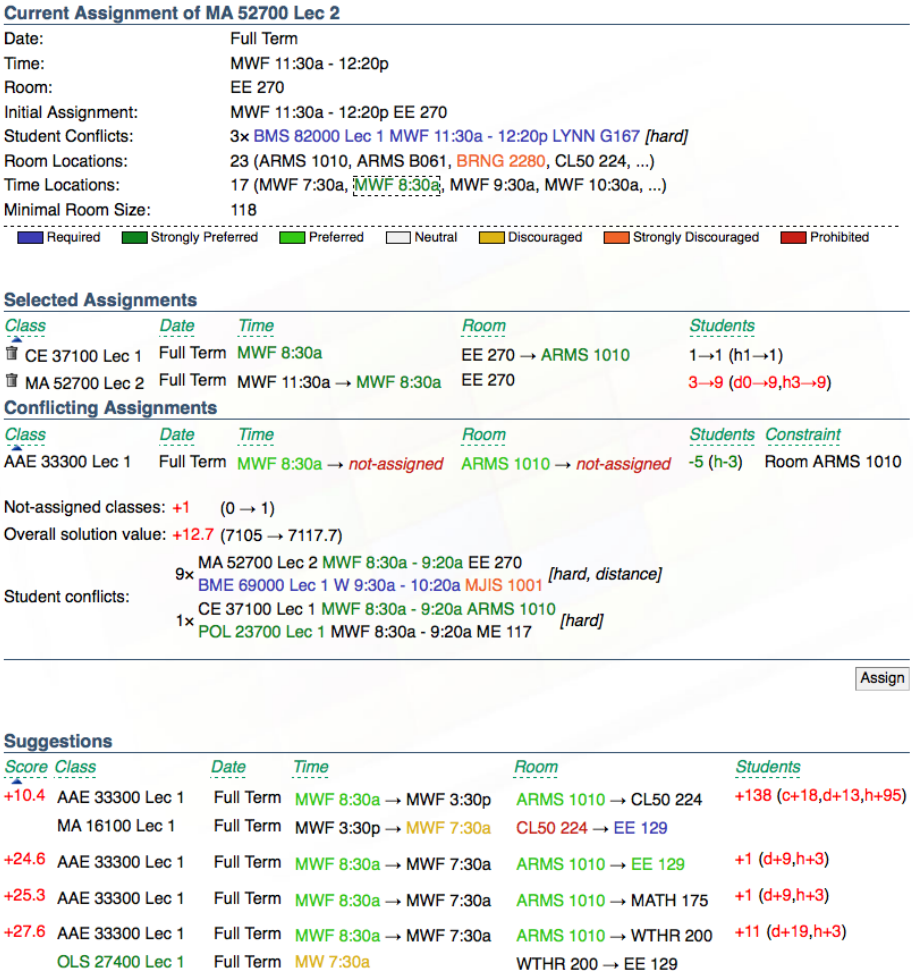

Fig. 1 Interactive solver interface for MA 52700 after selection of a new time assignment.

Figure 2 provides a more formal description of the interaction process leading to a desired solution. It is assumed that a solution δ of the timetabling problem P was generated by a standard timetabling solver such as that discussed by Murray et al. [4]. These become inputs for interactions with a user who may wish to explore changing the assignment of a class v_{bb}. The interactive solver BB generates a set of suggestions Ω along with conflicting assignments γ resulting from changes in v_{bb} and from any earlier selected assignments μ. Each suggestion $\omega \in \Omega$ represents a set of assignments resolving conflicts in γ.

The user has many possible responses, represented by the status variable S. When the user is satisfied with one of the suggestions ω, (s)he may *commit* the solution and the interactive process terminates. Any previously selected assignments μ are projected along with the new suggestions ω into δ. Alternatively, the user may suggest a specific assignment v_{bb}/d with the help of *selectAssignment(d)* to explore consequences of this

assignment by addition of v_{bb}/d to selected assignments μ. There is also a choice *selectClass(c)* to consider changes to a different class c, which becomes the new v_{bb} variable for future cycles. Note that the user can choose class c either from $\gamma \cup \mu$ or from Ω since these are available through the GUI. Another possibility, *removeClass(c)* allows deletion of a class c from μ and continuing with the reduced set of previously selected assignments. Finally, the user may decide to *abort* the interaction without applying any of the changes in μ to the original solution. If needed, (s)he can continue the interaction process by starting another interaction, e.g., selecting a different variable v_{bb}.

procedure INTERACTION(P, δ, v_{bb})
  $\mu = \emptyset$
  while true do
    $(\Omega, \gamma) =$ BB(P, δ, μ, v_{bb})
    $S =$ COMMUNICATION$(P, \delta, \mu, v_{bb}, \gamma, \Omega)$
    case (S) *commit*(ω): $\delta =$ JOIN$(\delta, \mu \cup \omega)$; $\mu = \emptyset$; **return**
      selectAssignment(d): $\mu = \mu \cup \{v_{bb}/d\}$
      selectClass(c): $v_{bb} = c$ (only $c \in \gamma \cup \mu$ or c from Ω are available)
      removeClass(c): $\mu = \mu \backslash \{c/d_c\}$
      abort: **return**
    end case
  end while
end procedure

Fig. 2 Cycle of interactive solving.

The timetabling problem P can be described as a constraint optimization problem [2] with domain variables, their domains, and constraints restricting the values of variables $(V, \mathcal{D}, C_h \cup C_s)$. A solution δ is a complete assignment of variables satisfying all hard constraints C_h. The quality of a solution is evaluated based on violations of soft constraints C_s, e.g., with the help of weighted constraints expressing their importance. Interactive timetabling can also work with a partial consistent assignment ρ (satisfying constraints having all variables assigned) and extend it by selecting assignments for additional variables. When a user asks the solver BB for a solution to the problem for the variable v_{bb} and required assignments μ, the hard constraints C_h must be extended by the constraint $v_{bb} \neq d_o$ (if v_{bb}/d_o is present in ρ) and $v_{bb} = d$ (if required), and by the constraints $v_1 = d_1, \ldots v_k = d_k$ for $\mu = \{v_1/d_1, \ldots, v_k/d_k\}$.

The constraint optimization problem becomes an input for the interactive solver (function BB at Figure 2). The solver is based on the branch and bound algorithm [2] with some extensions. Most importantly, the depth of the search is limited to allow changes to only a small number of variables (typically two or three), since the goal is only to create a new assignment for the variable v_{bb}. This limit may be extended and a subsequent call of BB invoked if needed. Second, a limited number n of best suggestions Ω is given to the user (note that the optimization criterion is computed through violations of weighted constraints and the best suggestions have the lowest value of weighted violations). As soon as the solution quality within current branch is worse than the quality of n-th found suggestion, its exploration is interrupted. Proper ordering of values for the assigned variables is necessary as is common for many other search algorithms. Each value is evaluated by its contribution to the value of the objective function and the best value is always selected. Finally, the search has a limit on the time (typically 5 seconds) in which to output a solution.

3 Experiments

To demonstrate the functionality of the branch and bound solver, it was applied on the largest problems at Purdue University, pu-fal07-llr and pu-spr07-llr. Characteristics of both problems, along with data sets and solutions, may be found at http://www.unitime.org, where the university timetabling application including an online demo is also available. Input solutions represent timetables used for the Fall and Spring semesters in 2007. Importantly, these timetables were generated by human schedulers using an automated solver [4] along with the interactive timetabling methods described here.

Suggestions were generated by the branch and bound solver for each class that was not fixed in time and room by a hard constraint. Changes to the assignments of up to two additional classes were allowed. Results in Table 1 are compared for runs with a time limit of 5 seconds (third and fifth columns) and runs without any time limit (second and fourth columns). The average time spent by the interactive solver roughly corresponds to the number of backtracks made. With the time limit, the solver was able to compute optimal solutions in more than half of the cases, with the quality of the solutions in the remaining cases being very good. Note that it was possible for the branch and bound solver to improve on the quality of input timetables since the solutions included manual changes by users having additional personal priorities.

Table 1 Results for the interactive solver.

Problem	pu-fal07-llr		pu-spr07-llr	
Classes	891		803	
Classes fixed in time & room (%)	31.0		33.8	
Time limit (s)	–	5	–	5
Optimal suggestion found (%)	98.4	51.5	99.2	67.0
Number of backtracks	66367.9	2886.9	13949.1	2592
Time spent (s)	128.6	4.7	39.9	4.2
Improvements in objective function (%)	+1.1	+0.8	+0.9	+0.7

4 Conclusions

A formal description of the approach applied to solving interactive timetabling problems at Purdue University has been presented. This technique was found to be a necessity by schedule managers who needed to make adjustments to automatically generated timetables as a result of changed requirements or to accommodate their own preferences. This allows creation of timetables that are ultimately used by the university.

Acknowledgements Hana Rudová was supported by the Ministry of Education, Youth and Sports of the Czech Republic under the research intent No. 0021622419 and by the Grant Agency of the Czech Republic with grant No. 201/07/0205.

References

1. Hadrien Cambazard, Fabien Demazeau, Narendra Jussien, and Philippe David. Interactively solving school timetabling problems using extensions of constraint programming. In Edmund Burke and Michael Trick, editors, *Practice and Theory of Automated Timetabling V*, pages 190–207. Springer-Verlag LNCS 3616, 2005.
2. Rina Dechter. *Constraint Processing*. Morgan Kaufmann Publishers, 2003.
3. Tomáš Müller and Roman Barták. Interactive timetabling: Concepts, techniques, and practical results. In Edmund Burke and Patrick De Causmaecker, editors, *PATAT 2002 — Proceedings of the 4th international conference on the Practice And Theory of Automated Timetabling*, pages 58–72, 2002.
4. Keith Murray, Tomáš Müller, and Hana Rudová. Modeling and solution of a complex university course timetabling problem. In Edmund Burke and Hana Rudová, editors, *Practice and Theory of Automated Timetabling VI*, pages 189–209. Springer-Verlag LNCS 3867, 2007.
5. Sylvain Piechowiak, Jingxua Ma, and René Mandiau. An open interactive timetabling tool. In Edmund K. Burke and Michael Trick, editors, *Practice and Theory of Automated Timetabling V*, pages 34–50. Springer-Verlag LNCS 3616, 2005.