System Demonstration of Interactive Course Timetabling

Tomáˇ s M uller¨ · Keith Murray · Hana Rudová*

1 Introduction

This system demonstration presents an approach to interactive timetabling used by the UniTime university timetabling system. This application, which is publicly available under an open source license, has been successfully applied at Purdue University [8], a large public university (39,000 students) with a broad spectrum of programs at the undergraduate and graduate levels. The complete system includes course timetabling, examination timetabling, event management, and student sectioning.

UniTime has a completely web-based interface using the Enterprise Edition of Java (J2EE). Hibernate is used to persist data in an SQL-enabled relational database (e.g., MySQL or Oracle) and an XML interface can be used to tie the application with other systems used by a university. The course timetabling, examination timetabling and student sectioning problems are modeled as constraint satisfaction and optimization problems (CSOP) and solved using the solver library [3]. This constraint-based local search framework has also been successfully applied to the International Timetabling Competition 2007, where it was among the finalists in all three tracks and the winner of two [4].

A major goal of the system design has been to facilitate requests for changes in the timetable that inevitably occur. Interactive timetabling was previously explored in [7], concentrating on interactive removal of clashes. [1] investigated explanations in constraint programming to handle dynamic changes in the timetable. Generation of a timetable was interactively controlled by the user in [2]. Earlier work by the present authors also examined construction of a timetable using minimal changes to an initial solution [5]. The work pre-

* Hana Rudová is supported by the Ministry of Education, Youth and Sports of the Czech Republic under the research intent No. 0021622419.

T. M uller¨ · 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@unitime.org

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

sented here encompasses an interactive mode for exploring possible changes, and easily making them, which was also found to be necessary. While this system demonstration focuses on making interactive changes to the course timetable, a similar approach is also used within the application for examination timetabling and event scheduling. Furthermore, an interactive phase of student sectioning (referred to as online student sectioning [6]) is currently being developed.

2 **Interactive Timetabling**

At Purdue, the complete university timetabling problem has been decomposed into a series of subproblems solved at the academic department level. Although each subproblem is solved separately, each solution considers all of the other problems for which a timetable has already been created. This coordination across problems is especially important when making interactive modifications to an existing timetable that may impact several others.

The course timetabling user interface consist of two parts: a data entry portion and a course timetabling solver. Basic data related to rooms, instructors, and courses (including all constituent classes) are entered via a series of web forms along with any associated preferences or requirements. Once all data have been entered into the system, the timetabling solver is used to create an automated timetable for the given (sub)problem. Subsequently, most changes are made using interactive timetabling. An exception is when multiple changes are desired in the input data. In this case, a new timetable is built from scratch or by using the minimal perturbation solver [5] which creates a solution to the modified timetabling problem while trying to minimize changes between the original and the new solution.

The same constraint model is used for both automated and interactive timetabling, with the same objective function consisting of satisfaction of

– preferences on time and rooms,
– distribution preferences that can be put between two or more classes (e.g., same room, back-to-back, or precedence),
– student conflicts (i.e., students that are expected to take two classes that either overlap in time or are back-to-back in rooms that are too far apart),
– divergence from the original solution, expressed as the number of students affected by a time change (room changes are usually consider less harmful),

along with several less important criteria. During interactive timetabling, the solver does not make any decisions, It does, however, provide users with a set of feasible solutions (and their associated costs) that can be reached via a backtracking process of limited depth. The user then determines the best tradeoff between accommodating a desired change and the costs imposed on the rest of the solution with a knowledge of what those costs will be. Moreover, to avoid a need for changing the input data, some of the hard constraints can be relaxed in the interactive mode. This means, for instance, that the user can put a class into a room different from the ones that were initially required. This is accomplished by making these hard constraints soft, but with too large of penalty imposed for the solver to suggest a change violating the constraint.

The timetabling user interface contains a set of pages that display various aspects of the current timetable. The user can view the classes in a time-resource grid for each resource (room, instructor, etc.), a list of assigned classes, or a list of yet-to-be-assigned classes. Changes to preferences or requirements made between the original and the current timetable, a history of the changes made using the interactive solver (which can also be used to easily undo such a change if needed), and various reports displaying room utilization, student conflicts, and violation of other soft constraints are also available.

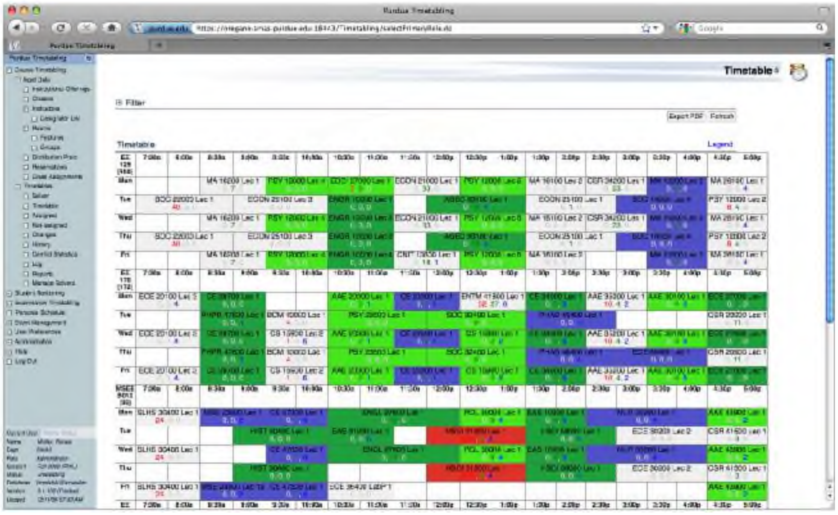

Fig. 1 Display of a timetable for given rooms.

A consistent color coding is used throughout the application. Required times or rooms are marked in blue, prohibited in red, light green and dark green for preferred and strongly preferred, yellow and orange, respectively, for discouraged and strongly discouraged. Figure 1 illustrates the timetable display provided for several rooms. The user can modify the assignment of one or more classes by clicking on a class in any of these views.

Each user interaction with the timetable can be seen as a sequence of changes to individual classes (beginning with the user selecting a class that needs to be changed). Figure 2 illustrates the information available to the user during each step of considering changes to the selected class of interest. The user may explore different options, consider various types of changes to the class, commit selected choices, or discard all changes considered. *Selected Assignments* describe changes already made to the timetable during the current interaction. *Conflicting Assignments* inform the user of any conflicts created in the timetable as a result of the selected assignments. *Suggestions* are optional changes the user may choose from that result in a feasible timetable. In each step (of an interaction), the user has the following possibilities:

– *Commit the change.* At this point, the actual timetable is modified, all the selected assignments are assigned and the conflicting classes, if any remain, are unassigned. The interaction is terminated.
– *Abandon the change.* The interaction is terminated without making any changes to the actual timetable.
– *Select a suggestion.* One of the suggestions is selected by the user, it is displayed together with the selected assignments. The user can still select another suggestion or try assign the class manually in the next step, e.g., if he or she is not satisfied with the resultant quality of the solution.

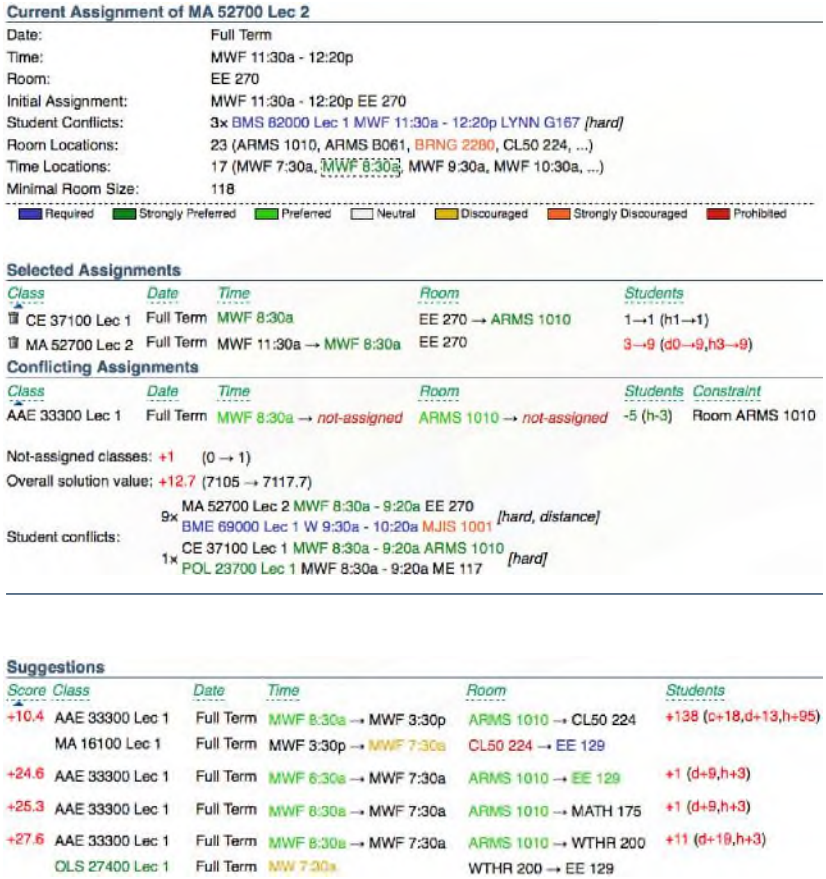

Fig. 2 Interactive solver interface for MA 52700 after selection of a new time assignment.

– *Select a placement.* Instead of choosing a suggestion, user picks a time and/or a room manually (for the selected class). This assignment is added into the list of selected assignments and the list of conflicts is recomputed together with the suggestions. The user may choose a different placement in the next step, e.g., if there are too many classes conflicting with the new assignment.
– *Remove a selected assignment.* An assignment is removed from the list of selected assignments, conflicts and suggestions are recomputed.
– *Select another class.* A different class (e.g., one of the conflicting classes) is selected. Suggestions are recomputed to include the selected class.

Besides the above actions, the user has a wide variety of additional choices that may help to find a desired change. For instance, the list of available suggestions can be filtered by additional criteria or the number of allowed additional changes of the provided suggestions can be increased.

Suggestions are computed using a branch and bound algorithm of a limited depth (initially allowing 2 additional changes). This process starts from the list of selected assignments, trying all possible placements for the selected class and resolving hard conflicts created by these changes. Only solutions that do not violate any hard constraints are allowed, which along with the quality of the *n*-th best solution found (*n* being the number of suggestions to be displayed) bounds the search. A more formal description of the algorithm is available in [8]. It is also shown here, based on comparison runs with no time limit on the branch and bound solutions, that a 5 second time limit is sufficient to present an optimal suggestion incorporating up to two additional changes in more than half of the cases (for the timetabling problem at Purdue). This time limit is important to keep the user interface interactive; however, when the limit is reached, the user has the option (e.g., when no desired suggestion is found) to recompute these suggestions with an increased time limit.

3 Conclusion

An extension of the UniTime course timetabling application has been presented that allows users to modify automatically computed timetables when a small number of changes are necessary to accommodate new parameters added after a timetable has been published. This interactive component allows the user to to find high quality options for meeting the additional problem parameters and deciding whether to modify the previous solution. The presented application is publicly available under an Open Source license [1], and can be downloaded from the UniTime web site http://www.unitime.org. This site also contains information about ongoing research, online documentation for the described system, and various real-life benchmark data sets for course timetabling, examination timetabling and student sectioning problems.

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. Hans-Joachim Goltz and Dirk Matzke. Combined interactive and automatic timetabling. In *Proceedings of the Practical Application of Constraint Technology and Logic Programming*, pages 529–535. The Practical Application Company Ltd, 1999.
3. Tomás ˇ Mu¨ller. *Constraint-based Timetabling*. PhD thesis, Charles University in Prague, Faculty of Mathematics and Physics, 2005.
4. Tomás ˇ M¨uller. ITC2007 solver description: a hybrid approach. *Annals of Operations Research*, 127:429– 446, 2009.
5. Tomás ˇ M¨uller, Roman Barták, and Hana Rudová. Minimal perturbation problem in course timetabling. In Edmund Burke and Michael Trick, editors, *Practice and Theory of Automated Timetabling, Selected Revised Papers*, pages 126–146. Springer-Verlag LNCS 3616, 2005.
6. Tomás ˇ M¨uller and Keith Murray. Comprehensive approach to student sectioning. In *PATAT 2008–Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling*, Montreal, Canada, 2008.
7. Sylvain Piechowiak, Jingxua Ma, and René Mandiau. An open interactive timetabling tool. In Edmund Burke and Michael Trick, editors, *Practice and Theory ofAutomated Timetabling V*, pages 34–50. Springer-Verlag LNCS 3616, 2005.
8. Hana Rudová, Tomás ˇ M¨uller, and Keith Murray. Complex university course timetabling. *Journal of Scheduling*, 2010. To appear.

[1] Constraint-based solver, including course timetabling, examination timetabling and student sectioning extensions is available under GNU Lesser General Public License (LGPL), the complete timetabling application is available under GNU General Public License (GPL).