Automated System for University Timetabling

Keith Murray and Tomáš Müller

Purdue University, West Lafayette IN 47907, USA
kmurray@purdue.edu
muller@purdue.edu

1 Introduction

Although university course timetabling is a widely studied topic, the use of automated timetabling systems is not widespread among large universities. This is particularly true in the United States, where the state of the art is typically to roll forward the last like semester's timetable and make adjustments to room assignments. University timetabling is a hard problem because of its size and the complexity of constraints needed to satisfy the demands of students and instructors. The problem is made harder yet by the need to develop a system that is easy for everyone involved in the process to use and understand, and for them to be satisfied with the results.

The system described here, and currently being implementation by Purdue University, successfully deals both with the issue of solving a large-scale problem and with addressing many of the human factors required in real applications.

The size of the problem (9,000 classes, 570 rooms, and 39,000 students with 259,000 class requests) has been made more manageable by decomposing it into a large lecture problem, consisting of centrally scheduled classes serving students in many disciplines, and multiple departmental problems. This partitioning also addresses the need to give departmental timetablers ownership of the process, which is important in a complex organization. Departmental timetablers have invaluable knowledge about what is and is not important in a solution that would be extremely difficult to incorporate into a black box solver. It is important to be able to leverage this knowledge with tools that can help sort through all of the constraints and costs to find solutions that satisfy user needs. The primary design goal was therefore to assist academic timetablers with the problem of building a good timetable, not necessarily finding a true optimal solution.

2 System Architecture

The system has been designed with a completely web-based interface (see Fig. 1) using the Enterprise Edition of Java 2 (J2EE), Hibernate, and Oracle Database.

The solver is based on an iterative forward search algorithm [3, 5]. This algorithm is similar to local search methods; however, in contrast to classical local search techniques, it operates over feasible though not necessarily complete

E. K. Burke, H. Rudová (Eds.): PATAT 2006, pp. 536–541. ISBN 80-210-3726-1.

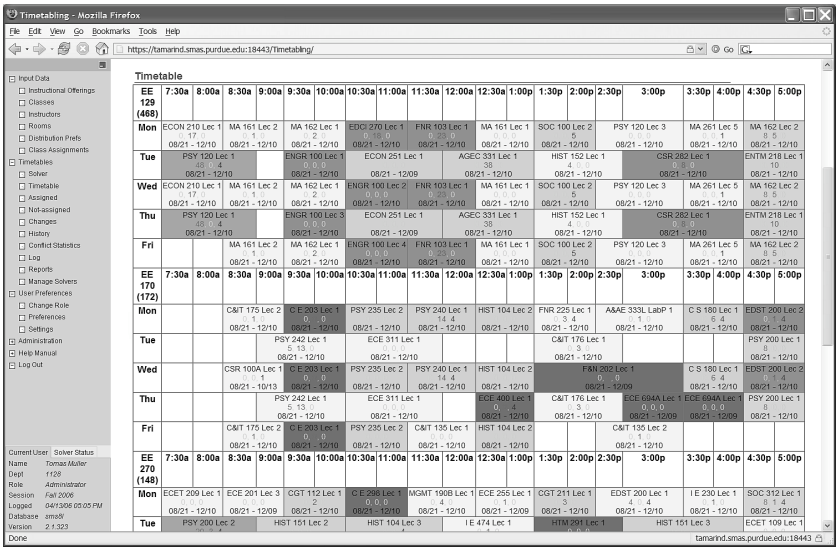

Fig. 1. Screen displaying timetable generated by solver.

solutions. In such a solution, some classes may be left unassigned. Still, all hard constraints on assigned classes must be satisfied. Such solutions are also easier to visualize and more meaningful to human users than complete but infeasible solutions. Because of the iterative character of the algorithm, the solver can easily start, stop, or continue from any feasible solution, either complete or incomplete. Moreover, the algorithm is able to cover dynamic aspects of the minimal perturbation problem [1, 5], allowing the number of changes to the solution (perturbations) to be kept as small as possible.

3 Critical Aspects of Application

In the course of developing a system that is useable in practice, it was necessary to confront a number of issues that are not typically addressed in the literature on timetabling, but which are critical to successful implementation. These included issues of the "fairness" of the solution across all departments with classes being timetabled, ability to check and resolve inconsistencies in input data, ease of introducing changes after a solution has been generated, creating and managing constraints and other data, and dealing with incomplete demand information for classes.

3.1 Competitive Behavior

A complicating aspect of real problems in educational timetabling is that there is competition for preferred times and rooms. Hard and soft constraints placed

on the problem are often reflective of this competitive behavior (e.g., limited instructor time availability, restrictive room requirements).

Hard constraints limit the solution space of the problem to reflect the needs or desires of those who place them. Soft constraints introduce costs into the objective function when violated. In either case, the more constraints placed on the problem by a particular class, instructor, or class offering department, the greater influence they will have on the solution. The general effect is to weight the solution in the favor of those who most heavily constrain the problem. This can create both harder problems to solve and solutions that are perceived as unfair by other affected groups or individuals. Inequity in the quality of time and room assignments received by different departments and faculty members doomed a previous attempt at automating the timetabling process at Purdue [2].

To counteract the tendency of the solution to favor those who place the most restrictions, a number of market leveling techniques were employed while modeling and solving the problem. The first was to weight the value of time preferences inversely proportional to the amount of time affected. A class with few restrictions on the times it may be taught has those restrictions more heavily weighted than a class with many restrictions. The intent is to make the total weight of all time restrictions on any class roughly equal. A second technique used in the solver was to introduce a balancing constraint. This is a semi-hard constraint in that it initially requires the classes offered by each department to be spread equitably across all times available for the class, but is automatically relaxed to become a cost penalty for poorly distributing time assignments if the desired distribution is overly constraining. Addressing this aspect of the real world problem was a key component of gaining user acceptance.

3.2 Interactive Changes

While it was known early that it would be necessary to deal with changes after an initial solution was found, it became clear the first time the system was used in practice that an interactive mode for exploring the possibility of changes, and easily making them, would be necessary. Following the original design philosophy of wanting to minimize the number of changes needed to a solution [1, 5], an approach was developed to present all feasible solutions and their costs that can be reached via a backtracking process of limited depth. The user is allowed to make the determination of 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. A further refinement was to allow some of the hard constraints to be relaxed in this mode. This means, for instance, that the user can put a class into a room different from the ones that were initially required.

Figure 2 shows a list of suggestions (nearby feasible solutions) provided to the user for a given class. The user may either pick one of these alternative solutions, ask the solver to provide additional suggestions by increasing the search depth (only two changes are allowed by default), or assign the class manually by selecting one of its possible placements. In this last case, a list of conflicting classes is shown together with a list of suggestions for resolving these conflicts.

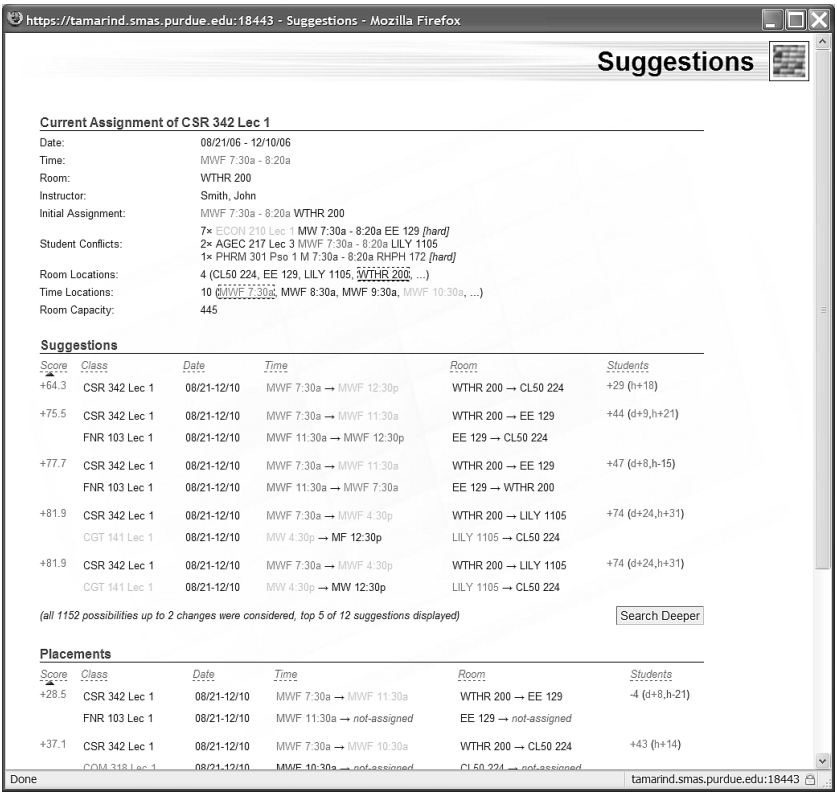

Fig. 2. Window displaying current assignment and suggested alternative assignments.

The user may either apply the selected assignment (which will cause all the conflicting classes to be unassigned), pick one of the suggestions, or start resolving the conflicts manually by selecting a new placement for one of the conflicting classes. This process can continue until all conflicts are resolved manually or a suggestion resolving all the remaining conflicts is found.

3.3 Data Consistency

Very often, especially during an early stage of the timetabling process, the input data provided by timetablers are inconsistent. This means that the problem is over-constrained, without any complete feasible solution. A very important aspect of the timetabling system is therefore an ability to provide enough information back to the timetablers describing why the solver is not able to find a complete solution.

In prior work on this problem [4, 5], a learning technique, called conflict-based statistics, was developed that helps the solver to escape from a local optimum.

This helps to avoid repetitive, unsuitable assignments to a class. In particular, conflicts caused by a past assignment, along with the assignment that caused them, are stored in memory. This learned information gathered during the search is also highly useful in providing the user with relevant data about inconsistencies and for highlighting difficult situations occurring in the problem.

3.4 Data Management

A major requirement for making the system usable across campus was ease of managing data on the classes to be timetabled and using that data for other existing processes. This led to a redesign of the timetabling database from one focused on a set of classes needing time and room assignments to one that better reflected the structure of various instructional offerings, with relationships between lectures, labs, etc. This was particularly valuable for departments with many offerings of the same class. The course structure could then be used to set constraints on large numbers of related classes (see Fig. 3).

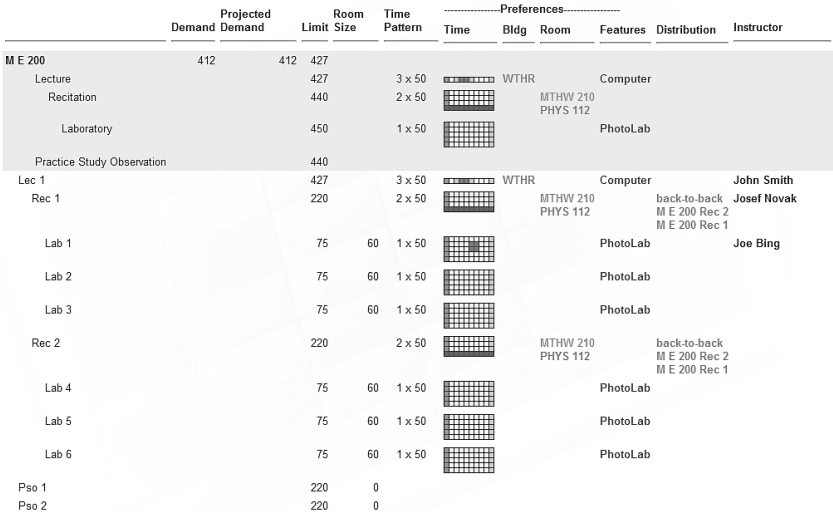

Fig. 3. Instructional offerings contain component classes in a logical structure reflecting the relationship among these classes. Constraints may be set on individual classes or on sets of classes of the same instructional type.

3.5 Student Demand and Sectioning

The primary optimization criterion in this problem is minimizing the number of conflicts between classes that are selected by students. Other preferences are

weighted against the number of student conflicts they may cause. Since demand data is not available for all students at the time the timetable must be created (e.g., specific course selections of incoming first year students are not know at the time the fall timetable is built), it is also necessary to incorporate projected information on student course selections into the joint demand matrix for classes. This complicates the initial sectioning process and requires additional algorithms for sectioning new students consistent with the best solution that has been found.

4 Conclusions

The system demonstrated here provides a complete solution to the course timetabling problem at Purdue University. It contains an attractive, intuitive user interface along with a solver that can be used in a variety of modes, ranging from a fully automated solution to assisting with manual assignments. Currently, the system is used by the central scheduling office for the large lecture timetabling problem. Initial use by departmental timetablers will begin for the spring 2007 term, with full distribution in time for planning the fall 2007 term.

From testing done on the large lecture problem (800 classes, 50 rooms, 86,000 class requests), the solver was proved to be able to stably provide better solutions than the previous manual solutions. For fall 2005 (last semester for which a manual solution to the large lecture problem was constructed), the solver was able to provide complete feasible solutions with approximately 1.2% more satisfied class requests (i.e., about 1000 class requests), leaving fewer than 0.6% class requests violated. It was also able to satisfy more preferences on time and space. Finally, it takes approximately 10 minutes for the solver to come up with a complete high quality solution, which is a significant improvement over a week of manual work.

References

[1] Roman Barták, Tomáš Muller, and Hana Rudová. A new approach to modeling and solving minimal perturbation problems. In *Recent Advances in Constraints*, pages 233–249. Springer Verlag LNAI 3010, 2004.

[2] Edward L. Mooney, Ronald L. Rardin, and W.J. Parmenter. Large scale classroom scheduling. *IIE Transactions*, 28:369–378, 1996.

[3] Tomáš Muller. *Constraint-based Timetabling*. PhD thesis, Charles University in Prague, Faculty of Mathematics and Physics, 2005.

[4] Tomáš Muller, Roman Barták, and Hana Rudová. Conflict-based statistics. In *EU/ME Workshop on Design and Evaluation of Advanced Hybrid Meta-Heuristics*. 2004.

[5] H. Rudová T. Müller, R. Barták. 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.