University Course Timetabling & Student Sectioning System

Tomáš Müller, Keith Murray, Stephanie Schluttenhofer
muller@unitime.org, kmurray@unitime.org, says@unitime.org
Space Management and Academic Scheduling, Purdue University
400 Centennial Mall Drive, West Lafayette, IN 47907-2016, USA

Abstract

An enterprise system for automated construction of course timetables and student schedules is presented and discussed. This is a distributed system that allows multiple university and departmental schedule managers to coordinate efforts to build and modify a course schedule that meets their diverse organizational needs while allowing for minimization of student course conflicts. Creation and modification of individual student schedules is also addressed.

Although the university course timetabling problem has received considerable attention, and many potentially useful solution approaches have been proposed, there have been relatively few successful applications to actual university problems of a large scale (Carter & Laporte 1998; Petrovic & Burke 2004; McCollum 2006). One such system that has recently been successfully implemented is discussed here, along with proposed approaches for sectioning individual students. The initial creation of a timetable and student schedules are addressed, along with interactive changes to each.

The system described here was developed to create and modify course timetables that better meet student course demand and allow students to be assigned to the constituent course sections in a way that minimizes conflicts (Murray, Müller, & Rudová 2007). The initial implementation has been at Purdue University, which is a large public university (39,000 students) with a broad spectrum of programs at the undergraduate and graduate levels. In a typical term there are 9,000 classes offered using 570 teaching spaces. Approximately 259,000 individual student class requests must be satisfied.

The complete university timetabling problem is decomposed into a series of subproblems to be solved at the academic department level. Several other special problems, where shared resources or student interactions are of critical importance, are solved institution wide. A major consideration in designing the system has been how to best support this distributed construction of departmental timetables while providing central coordination of the overall problem.

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

System Architecture

The system has been designed with 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). The system architecture consists of an enterprise web-based application and a constraint solver engine (see Fig. 1). The Web Server contains the necessary persistence, business and presentation logic, including XML interfaces for communication with other systems. Course timetabling and student sectioning problems are solved by the Solver Server. Since timetabling is done at the departmental level (each department builds its own timetable), the Solver Server needs to be able to work with multiple problem instances at the same time. Also, if needed, the load can be spread between multiple Solver Servers. Each problem is modeled as a constraint satisfaction and optimization problem (CSOP) and solved using the iterative forward search algorithm (Müller 2005). This functionality is provided by the Constraint Solver library.

The course timetabling portion of the application has been used university-wide at Purdue since January 2007. The solver has been used for several sub-problems beginning in 2005. Student sectioning is currently under development. Exam scheduling and event management capabilities are planned for inclusion in the future.

The presented application is publicly available under an Open Source license[1], and it 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 and student sectioning problems.

[1]Constraint-based solver, including course 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).

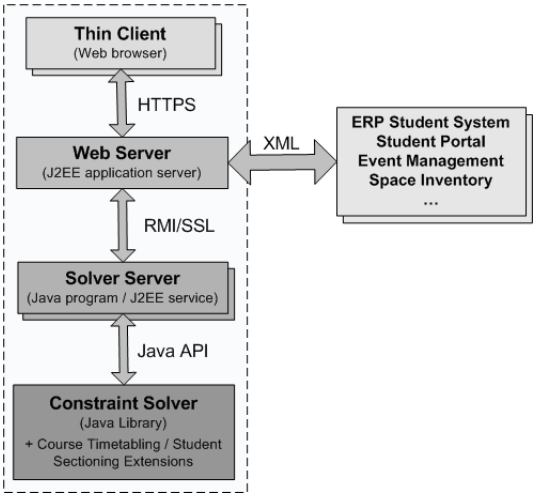
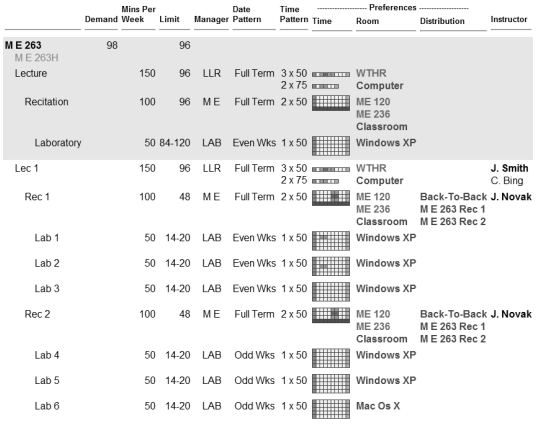


Figure 1: System architecture

User Interface

A major requirement for making the system usable across a university is the ability to represent the wide variety of course structures and conveniently manage data on all the classes to be timetabled. To achieve this, a course model was developed that allows the structure of all instructional offerings, including relationships between lectures, labs, etc. to be reflected in the database as individual classes and a series of constraints between them. The ability to work with an interface that recognizes the structure of each course is particularly valuable for departments with many classes of the same offering. The structure can then be used to set constraints on large numbers of related classes (see Fig. 2).



	Demand	Mins Per Week	Limit	Manager	Date Pattern	Time Pattern	Preferences: Time	Preferences: Room	Preferences: Distribution	Instructor
M E 263	98		96							
M E 263H										
Lecture		150	96	LLR	Full Term	3 x 50 2 x 75		WTHR Computer		
Recitation		100	96	M E	Full Term	2 x 50		ME 120 ME 236 Classroom		
Laboratory		50	84-120	LAB	Even Wks	1 x 50		Windows XP		
Lec 1		150	96	LLR	Full Term	3 x 50 2 x 75		WTHR Computer		J. Smith C. Bing
Rec 1		100	48	M E	Full Term	2 x 50		ME 120 ME 236 Classroom	Back-To-Back M E 263 Rec 1 M E 263 Rec 2	J. Novak
Lab 1		50	14-20	LAB	Even Wks	1 x 50		Windows XP		
Lab 2		50	14-20	LAB	Even Wks	1 x 50		Windows XP		
Lab 3		50	14-20	LAB	Even Wks	1 x 50		Windows XP		
Rec 2		100	48	M E	Full Term	2 x 50		ME 120 ME 236 Classroom	Back-To-Back M E 263 Rec 1 M E 263 Rec 2	J. Novak
Lab 4		50	14-20	LAB	Odd Wks	1 x 50		Windows XP		
Lab 5		50	14-20	LAB	Odd Wks	1 x 50		Windows XP		
Lab 6		50	14-20	LAB	Odd Wks	1 x 50		Mac Os X		

Figure 2: Instructional offering input

Another important aspect of this application is the support for distributed responsibilities over the timetabling process. Being able to individually timetable each department is required because departmental timetablers have a intimate knowledge of the needs of the courses offered, the faculty who might be able to teach a particular class, and the spaces available for specialized instructions. Maintaining each department's sense of ownership in the timetables that are produced is also an important factor in their acceptance of the solutions produced by an automated timetabling process.

The user interface for the solver is another very important part of the application. A user is able to produce and store as many timetables as desired; however, only one solution can be committed for each problem. The application also provides many tools for evaluating a timetable, compare two timetables, finding inconsistencies or potential problems in the input data, and to make manual changes to a timetable. Figure 3 illustrates the display of a timetable produced by the system. Satisfaction of a particular types of preferences (room, time, student conflicts, etc.) is visualized by the background of each class. The system also provides all the necessary coordination between departmental timetables, which is important especially between departments that share resources (instructors, rooms, students, etc.).

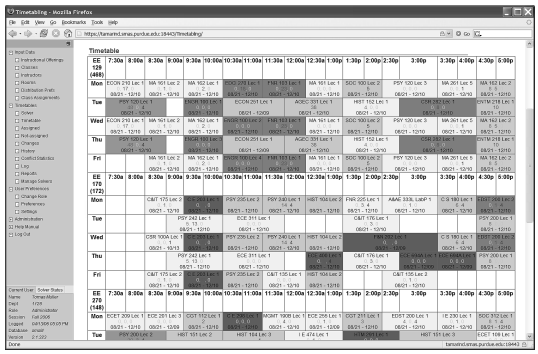


Figure 3: Timetable display

Solver

The solver is based on the iterative forward search algorithm (Müller 2005). This algorithm is similar to local search methods; however, in contrast to classical local search techniques, it operates over feasible though not necessarily complete solutions. In such a solution, some classes may be left unassigned. Still, all constraints on assigned classes must be satisfied. Such solutions are 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 (Müller, Rudová, & Barták 2005), allowing the number of changes to the solution (perturbations) to be kept as small as possible.

Competitive Behavior A complicating aspect of real timetabling problems is the competition for preferred times and rooms. 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 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. This has been addressed through weighting of preferences and the introduction of an automatic balancing constraint.

Interactive Changes A major goal of the system design was to facilitate the multiple requests for changes in the timetable that inevitably occur. While initially an approach for finding alternate solutions with minimal change to the initial solution was incorporated, an interactive mode for exploring the possibility of changes, and easily making them, was also found to be necessary. An approach was therefore 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.

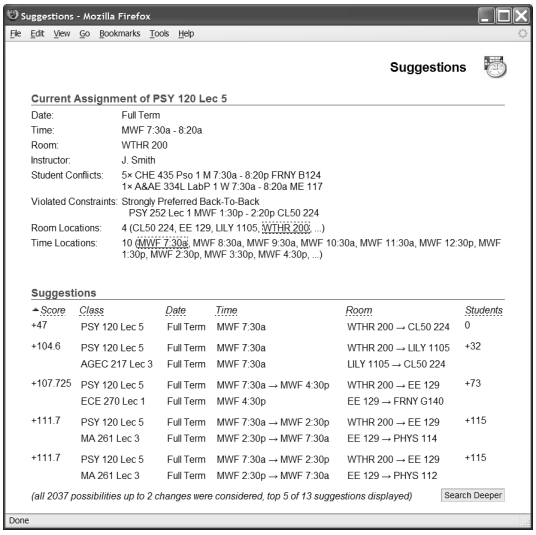


Figure 4: Suggestions provided to the user

Figure 4 shows a list of suggestions (nearby feasible solutions) provided to the user for the given class. A user can either pick one of them, ask solver to provide more suggestions by increasing the search depth (only two changes are allowed by default) or try to assign a 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. The user can either apply the selected assignment (which will cause the conflicting classes to be unassigned), pick one of the suggestions, or manually resolve conflicts by selecting a new placement for each conflicting class.

Data Consistency Often during the early stages of the timetabling process, the input data provided by schedule managers 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 (Müller, Rudová, & Barták 2005), 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 information learned during the search is also useful in providing the user with relevant information about inconsistencies and for highlighting difficulties in the problem.

Student Sectioning

Before and during the construction of the timetable, course demand can be collected from students. During this pre-registration process, each student can create a list of requested courses together with his or her preferences (see Fig. 5). These preferences contain course priorities (order of courses based on their importance for the student), alternative course requests, free time requests, wait-list preferences (if space is not available, should he or she be assigned to the appropriate wait-list for the course), and additional schedule distribution preferences.

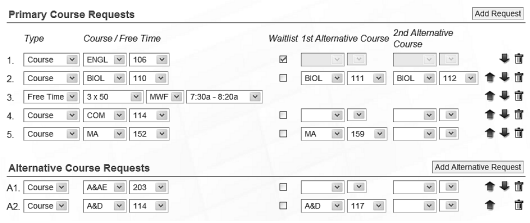


Figure 5: Student course demands example

Initial Sectioning During the construction of the course timetable, course demands of pre-registered students are considered. Since many students are anticipated to register later in the process, projected course demands are considered as well. These are deducted from the last-like semester enrollments, e.g., fall 2006 course enrollment patterns are used to predict fall 2007 course demands. Minimization of potential student conflicts is one of the optimization criteria of the timetabling solver. Two classes are conflicting, i.e., they cannot be attended by the same students, if they are overlapping in time or if they are back-to-back (the second class starts just after the first ends) and placed in rooms that are too far apart.

Before the course timetabling solver is started, an initial sectioning of students into classes is processed. However, it is still possible to improve on the number of student conflicts in the solution. This is accomplished by moving students between alternative classes of the same course during or after the search for a timetabling solution.

Batch Sectioning After the course timetable for the entire university is constructed, the batch student sectioning process is executed. In this process, all pre-registered students are assigned to specific sections (classes) of courses in order to minimize conflicts as well as optimize preferences provided by students. Additional constraints deducted from the course structure and reservations on space in particular courses or classes are respected. Students who are not able to enroll in a requested course (or alternate) are enrolled to the appropriate wait-lists.

The batch student sectioning also uses the projected student demand to compute the expected number of students in each class for the subsequent online sectioning phase. Pre-registered students take precedence over projected student demand however. This means that a pre-registered student cannot be bumped out a requested course because of a projected student, but he or she may be assigned to a class at a time that does not prevent projected students from taking the course as well. Based on the computed solution, pre-registered students are assigned to classes and wait-lists and the projected students course demands are used to identify space in each section that is to be reserved for students that are not yet registered. This information is then used in the online sectioning phase in order to direct students away from sections that are expected to be taken by later enrolling students.

Online Sectioning After batch sectioning takes place, students can make changes in their schedules using the online interface. During this phase, pre-registered students are allowed to remove themselves from requested courses or request additional courses and a new sectioning solution is provided in real-time. They can also change their class enrollments if there are other classes of the course that are available or wait-list themselves to classes that are not currently available. Wait-lists are automatically processed as space is freed in courses and classes. Some changes in the course timetable may also occur, potentiality causing some re-sectioning of enrolled students. New students use the same interface as pre-registered students. They begin by entering course requests, based on which they are sectioned to classes in real-time. They may then continue in the on-line sectioning the same as continuing students (see Fig. 6).

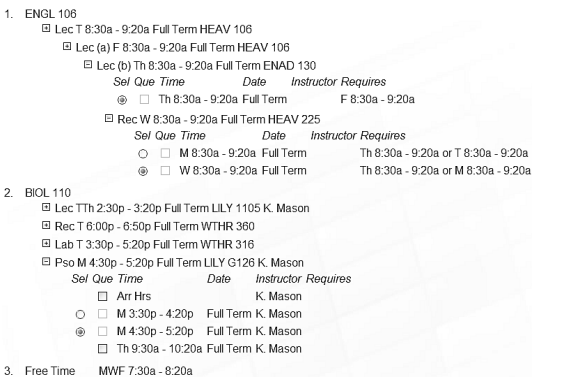


Figure 6: Student class enrollments with possible choices

As students submit schedule requests, each course is ranked in priority order. During real-time student sectioning, the search employs a backtracking process considering possible assignments beginning with those classes associated with the students highest priority course. As it evaluates each possible assignment, it compares available space with the space expected to be taken by the future students for each class. The difference between available space and the expected need for each class is used to direct students away from class assignments that would result in excess demand; however, in no case is an eligible student blocked from scheduling a course offering as a result of expected future demand. As students are assigned to specific classes during the sectioning process, the expected demand for each class is adjusted to reflect the assignment.

Conclusions

Based on the successful implementation of the system described here, it is clear that complex university course timetabling and student sectioning problems can be solved in practice. This system provides a method for modeling the structure of course offerings and other essential problem constraints. It uses these along with student demand to to build and modify course timetables and create individual student class schedules that satisfy student needs and minimize conflicts. The resulting application should be usable by a wide range of institutions wishing to address similar problems. Other universities wanting to explore these solution approaches are invited to make use of the application and/or participate in the continuing research in this area.

References

Carter, M. W., and Laporte, G. 1998. Recent developments in practical course timetabling. In Burke, E., and Carter, M., eds., *Practice and Theory of Automated Timetabling II*, 3–19. Springer-Verlag LNCS 1408.

McCollum, B. 2006. University timetabling: Bridging the gap. In Burke, E. K., and Rudová, H., eds., *PATAT 2006 — Proceedings of the 6th International Conference on the Practice and Theory of Automated Timetabling*, 15–35. Masaryk University.

Müller, T.; Rudová, H.; and Barták, R. 2005. Minimal perturbation problem in course timetabling. In Burke, E., and Trick, M., eds., *Practice and Theory of Automated Timetabling V*. Springer-Verlag LNCS 3616. 126–146.

Müller, T. 2005. *Constraint-based Timetabling*. Ph.D. Dissertation, Charles University in Prague, Faculty of Mathematics and Physics.

Murray, K.; Müller, T.; and Rudová, H. 2007. Modeling and solution of a complex university course timetabling problem. In Burke, E., and Rudová, H., eds., *Practice and Theory of Automated Timetabling VI*. Springer-Verlag. In print.

Petrovic, S., and Burke, E. K. 2004. University timetabling. In Leung, J. Y.-T., ed., *The Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. CRC Press. chapter 45.