Comprehensive Approach to Student Sectioning

Tomáš Müller and Keith Murray

Purdue University, West Lafayette IN 47907, USA
`muller@unitime.org`, `kmurray@unitime.org`

Abstract. Student sectioning is the problem of assigning students to particular sections of courses they request while respecting constraints such as course structures, section limits, and reserved spaces. Students may also provide preferences on class times and course alternatives. In this paper, three approaches to this problem are examined and combined in order to tackle it on a practical level: student sectioning during course timetabling, batch sectioning after a complete timetable is developed, and online sectioning for making additional changes to student schedules. An application and some practical results of the proposed solutions based on actual data are also included.

1 Introduction

Student sectioning is the problem of assigning students to classes (i.e., individual sections of a course) while respecting individual student course requests along with additional constraints (e.g., a student cannot attend two classes that overlap in time). The traditional reason for optimizing student sectioning, rather than students choosing individual classes, is to maximize the number of satisfied student course requests. In the on-line version of the problem this means maintaining a distribution of available space in classes across times needed to accommodate requests by students who come later in the process. To meet modern expectations, optimizing student sectioning must also consider other preferences and priorities of students while creating their class schedules. This is an important problem for institutions offering many courses with multiple sections.

Academic timetabling problems are frequently categorized as either school timetabling, course timetabling, or exam timetabling [17]. Student sectioning usually resides outside of this categorization since the task is not to allocate a set of events (e.g., classes or exams across time and space); however, some form of student sectioning may be involved in a demand-driven course timetabling process in order to minimize the number of potential student conflicts between classes [5, 12, 18]. In this case, student sectioning is necessary to define the number of students in common for each pair of classes.

In practice, it is not sufficient to consider student sectioning only as a subproblem of timetabling. Especially when the total demand for classes is not known while creating the timetable, it may be necessary to section additional students at a later time. In a distributed timetabling system such as used in this work [14], there may also be sectioning issues that arise (e.g., respecting preferences) while solving multiple sub-problems that involve the same students. This is particularly an issue when they are solved over a period of several weeks during which changes occur in class offerings and student demand. In an era where students expect 24/7 service, there is also a need to address schedule changes both efficiently and without jeopardizing the ability of latter enrolling students to receive courses they require.

A variety of strategies for sectioning students have been discussed in the literature on educational timetabling. Carter and Laporte [6] and Schaerf [17] provide a good overview of previous work on the sectioning problem. Aspects of several different approaches to sectioning are of interest when developing a system for constructing both course timetables and individual student schedules. Aubin and Ferland [9] considered the problem of sectioning, or grouping, students during the timetabling process. They generated an initial timetable with students assigned to sections, then applied an iterative process to adjust the timetable and student groupings successively until no further improvement was found. Hertz and Robert [8] adopted a similar approach, decomposing the course scheduling problem into a series of assignment type subproblems (timetabling, sectioning, and room assignment) that were iterated on while an increasing number of constraints were considered. Banks, et al. [2] used subset constraints between sections in a CSP formulation of the timetabling problem to avoid conflicts between sets of student course choices. Amintoosi and Haddadnia [1] proposed a fuzzy clustering algorithm to create an initial sectioning prior to timetabling a set of classes. The primary aim of considering sectioning in all of these cases was to improve the balance between class enrollments and to minimize conflicts preventing assignment of students to a full set of selected courses.

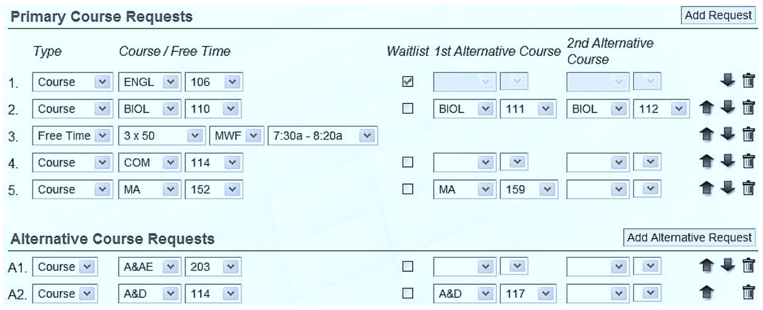
Student sectioning has also been considered as a separate problem when applied to a fixed timetable. Laporte and Desroches [10] implemented a branch and bound procedure that first assigned students to sections without considering section enrollments or room capacities and then iteratively modified these assignments to reduce the imbalance in section sizes and mismatches with assigned room sizes. Sabin and Winter [15] proposed a heuristic which assigned weights to each student's set of course choices to assess its complexity. Those with the greatest complexity were scheduled first.

The idea of accommodating student preferences during the sectioning process has also received attention. In an early work, Busam [4] presented an algorithm for assignment of students to classes in a fixed timetable that allowed for student section preferences, if consistent with the objective of balanced sections, by ordering assignments based on the number of students preferring each section. More recently, Feldman and Golumbic [7] introduced priorities on constraints in the student sectioning problem indicating which schedules were preferred over others by the student. A number of algorithms were then presented for finding the the best schedule minimizing violation of student priorities. Sampson and Weiss [16] furthered the idea of accommodating student preferences in both the timetabling and sectioning problems by introducing a heuristic approach based on each student's priority ordering of the courses and sections they wished to enroll in. The timetable and section assignments were built by an algorithm maximizing the weighted value of assignments made.

In this paper, student sectioning processes will be considered both as a component of the timetable construction process and as a means to optimize student preferences. The general student sectioning problem is first described in the following section. Next, a three phase approach to this problem is outlined which considers student class assignments during and after construction of the timetable. In the first phase, a course timetable is created while minimization of potential student conflicts is used as one of the optimization criteria. In the next phase, after a timetable for the whole university has been completed, all registered students are enrolled into sections of their requested courses. Projected student requests are used in this step in order to anticipate class needs of incoming, but not yet registered, students. In the final phase, all students are free to select or change class schedules using an online interface. During this process, students are sectioned in the order they access the system; however, the proposed technique avoids using all of the space available in sections at times that are expected to be required by students who will register later in the process (e.g., first year students and transfers). The last section of this paper is devoted to experimental results. Here, several evaluations of the proposed approach are presented based on actual data from Purdue University.

2 Problem Model

Course demand data may be collected from students in a format similar to that shown in Figure 1. Each student is able to create a list of requested courses in his or her order of priority. Additional preferences may also be included for alternative courses, free time requests, and wait-listing on courses.



Primary Course Requests

	Type	Course / Free Time			Waitlist	1st Alternative Course		2nd Alternative Course	
1.	Course	ENGL	106		☑				
2.	Course	BIOL	110		☐	BIOL	111	BIOL	112
3.	Free Time	3 x 50	MWF	7:30a - 8:20a					
4.	Course	COM	114		☐				
5.	Course	MA	152		☐	MA	159		

Alternative Course Requests

A1.	Course	A&AE	203	☐				
A2.	Course	A&D	114	☐	A&D	117		

Fig. 1. Student course demand example

In the example shown in Figure 1, the student wishes to attend four courses and have a free time block between 7:30 am and 8:30 am on Mondays, Wednesdays and Fridays. The ordering of the requests indicates their importance to the student. This means that if a student can attend only one of two requested courses, e.g., because the only available sections of these courses overlap in time, he should be enrolled into the course with a higher priority. The same principle applies to free times. In the example above, the student would still wish to attend the English or Biology courses (1st and 2nd course requests) even if they overlap with the requested free time (3rd request), however, he would not attend the Communication or Math course during that time (4th and 5th request).

At Purdue University most classes are offered during standard time blocks. For instance, a 3 hour (150 minutes excluding breaks) class is either offered as one three hour long meeting, two meetings of 75 minutes that take place on Tuesday and Thursday with the same start time, or three meetings on Monday, Wednesday, and Friday starting at the same time. Possible starting times are set so that time use is maximized, without leaving unnecessary empty time windows in the rooms. In the model, free time requests can also be specified using the same standardized time blocks. This way the amount of time blocked by each free time request roughly corresponds with amount of time needed by a course. If a student desires an entire morning free he or she would need to use multiple free time requests, lowering the priorities on subsequent course requests. This gives other students with fewer free time requirements a better chance of receiving requested courses since they carry a higher priority. There is thus an explicit trade-off between a higher probability of receiving desired courses or a higher probability of receiving desired free times. Each student is free to set priorities to fit his or her needs.

The wait-list preferences (Waitlist toggle on Figure 1) indicate whether a student wishes to be assigned to the appropriate wait-list for a course if he or she cannot be enrolled in it, e.g., because of limited space availability.

Each request for a course that a student does not wish to be on a wait-list for can have acceptable alternative courses designated on the same line. This means that if the student cannot be enrolled in the requested course, he or she can be enrolled in the listed alternative course instead. A list of general alternative course requests can be provided as well. These serve as alternatives to all provided course requests that are not wait-listed and can help the student to obtain a complete schedule with the desired number of courses. The intended use of these two types of alternatives is that a course meeting a similar requirement may be taken in place of a given course listed on the same line (a student may not be enrolled in both a course and its alternative), additional optional courses that help the student to create a complete schedule with the desired number of courses or credit hours may be listed in the Alternative Course Requests section.

The problem is modeled as a Constraint Satisfaction and Optimization Problem (CSOP). Each request by a student for a course (including any alternatives) or a free time is represented as a variable. An assignment to a course request is a list of sections of the course (or alternative) into which the student is to be enrolled. Some combinations of sections of a course may not be allowed by the defined course structure (e.g., the only allowable combinations of sections for a course with two lectures and four laboratories may be the first lecture taken with the first or second laboratory, or the second lecture taken with the third or fourth laboratory). An assignment of a free time request is the requested time block. A free time may be left unassigned in the case where the student needs to attend a section of a course with higher priority that overlaps with the free time.

The problem also consists of the following constraints:

- Each section has a limit on the number of students that can be enrolled in it. Some sections can be marked as unlimited.
- Each course has a course structure defining the valid combinations of sections into which a student can be enrolled, these sections cannot overlap in time.
- A student cannot attend two courses that have sections that are overlapping in time. He or she also cannot attend a section that is overlapping with an assigned free time request.
- Reservations can be defined on courses and/or sections. Each course reservation consists of a course, a list of students that meet the reservation criteria (e.g., students in a program of study the course is designed for) and the number of students that can be enrolled into the course using this reservation. Similarly, a section reservation consists of a section, a list of students, and the number of students that can be enrolled using this reservation.
- An alternative course request can have an assignment only if there is an unassigned (non-alternative) course request of the same student that is not wait-listed. Only one alternative course request can be assigned in place of each such unassigned (non-alternative) course request. Free time requests do not currently have any alternatives.
- A course or free time request can be left unassigned only when there is no available assignment for the request (i.e., each assignment violates the available limit of a section, a reservation, or it overlaps in time with an assigned request of higher priority by the same student).

A solution of the student sectioning problem is a most complete set of assignments of requests that meets the above constraints. Among these, the aim is to maximize the overall priority of the assignments together with minimizing the use of course alternatives provided by students (i.e., courses that are listed on the same line as the requested course on Figure 1.). Both of these objectives are modeled by the maximization of a single weight that is computed for each assignment of a request. Thus $weight(a \in dom(R)) = 0.9^{prior(R)} \times 0.5^{alt(a)}$, where $prior(R)$ is the priority of the requested course or free time, and $alt(a)$ is the ordering of an alternate assignment a to a request R. The value of a is zero if the student is enrolled into the requested course or free time, 1 if he/she is enrolled into the first provided alternative course, 2 if into the second provided alternative course, etc. In order to ensure that general alternative course requests (i.e., courses listed in the Alternative Course Requests section of Figure 1.) will have a lower weight than normal requests, the priority assigned to each alternative course requests is increased by the number of ordinary requests (e.g., request A1 from Figure 1 has priority of 6, A2 has priority of 7).

The second criteria that is being considered is minimization of the overall number of distance student conflicts. A distance student conflict occurs between two sections attended by the student that are back-to-back in time and located in rooms that are too far a part. In these tests, the limit on distance between two rooms is set to 670 meters for sections having a break of interval of 10 minutes or less between, 1000 meters if there are more than 10 minutes but less then 20 minutes between. There are no distance conflicts between sections that have an interval of more than 20 minutes between them.

3 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 a major 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.

This initial sectioning approach and its application to Purdue University course timetabling problem is discussed in more details in earlier work [14].

4 Batch Sectioning

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

This batch sectioning process also makes use of the projected student demand to compute an expected number of students requiring 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 in favor 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 student course demands are used to identify space in each section to be reserved for students that are not yet registered. This information is used in the online sectioning phase to direct students away from sections that are expected to be taken by later enrolling students.

4.1 Batch Sectioning Algorithm

Batch sectioning is implemented using the Constraint Solver Library initially developed for solution of the timetabling problem [11]. It is based on the iterative forward search algorithm [12]. 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 these solutions some variables may be left unassigned, but all hard constraints must be satisfied.

The algorithm works in phases during which it selects potential assignments from one of the six neighborhoods described below. When there are no more assignments to select within the current neighborhood, the search progresses to the next neighborhood. The solver starts with ($\mathbb{N}_{B\&B}$), and proceeds through neighborhoods in the order listed. After the last neighborhood ($\mathbb{N}_{Resect}$) is reached and exhausted, the search returns to ($\mathbb{N}_{Swap}$) and continues in this manner. The search is stopped when a set time limit is reached.

$\mathbb{N}_{B\&B}$: All students are taken in order determined by the average number of section choices available for the courses they have selected (students with fewer choices first). A branch & bound technique is used to evaluate the best possible assignment of each student to available classes. Assignments made to students previously sectioned are not changed. Course and free time requests are considered in order based on the students' priorities, and the search is bounded by the best schedule found so far. This phase is similar to the online sectioning algorithm discussed at greater length in Section 5.1.

$\mathbb{N}_{Swap}$: All students who do not have a complete schedule (i.e., are not enrolled in the desired number of courses) are taken in random order. For each of these students, all unassigned course and free time requests are considered, and all possible section assignments are evaluated. The search looks for an improving assignment that has either no conflict with other assigned requests, or where any conflicting requests can be reassigned with alternative non-conflicting assignments. Conflicting requests include other assignments of the student that overlap in time with the new assignment, or assignments of other students that must be changed in order to satisfy the new assignment (e.g., due to a section limit). Among all possible new assignments, the one which improves the overall objective the most is selected along with the computed reassignments.

$\mathbb{N}_{Ifs}$: Selections of a variable and an associated value are used for a number of iterations proportional to the number of unassigned requests. In each step, an unassigned request is first selected randomly. For each such request, all possible enrollments are enumerated and the best one is selected and assigned. This may cause other conflicting requests to be unassigned. If there is a choice of how to resolve a conflict (e.g., what request to unassign in order to free a space in a section), a request with lowest weight is always selected. The quality of a selected assignment is computed using a weighted sum of the optimization criteria including the number of assignments that need to be unassigned in order to assign the selected value. Conflict-based Statistics [13] is used during this process to prevent repetitive assignments of the same values by memorizing conflicts which have occurred during the search together with their frequency and the assignments that caused them.

$\mathbb{N}_{Bt}$: Both assigned and unassigned requests are considered in this phase. For each (taken in random order) an attempt is made to find an improving assignment (including potential reassignments of other requests) using a backtracking technique of limited depth that tries to resolve conflicting assignments. The depth limit is usually set to three, which means that three other requests can be reassigned together with the new assignment of the selected request. Unlike the previous neighborhood, no conflicting assignment can be left unassigned, and the search is bounded by the value of the current assignment and/or the best improvement found so far.

$\mathbb{N}_{Un}$: All requests by a student (or group of students) are unassigned. Students are selected randomly, however, problematic students are selected with a higher probability. A student is identified as problematic if a request by that student is preventing some other student from enrolling in a course but the $\mathbb{N}_{Swap}$ was unable to find a non-conflicting reassignment. After an unassignment of a student, the same neighbourhood is used with a probability of 90%, otherwise the search moves to the next neighborhood.

$\mathbb{N}_{Resect}$: The same technique as in $\mathbb{N}_{B\&B}$ is used to reassign students without a complete schedule. The students are taken in random order, however, all students that were unassigned in the previous step are considered only after all other students are processed. Once done, the search continues with $\mathbb{N}_{Swap}$ neighborhood.

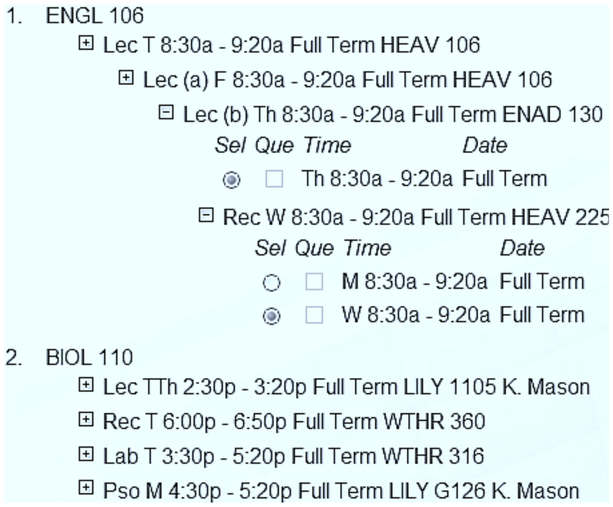
When both real student requests (i.e., requests from students who have preregistered) and projected student requests (i.e., requests estimated based on last-like term student enrollments) are sectioned together, the first cycle of the above algorithm is only carried out on the real student requests. This ensures that no real student request is left unassigned because of projected requests. A second cycle is then carried out with projected requests using the $\mathbb{N}_{B\&B}$ neighborhood on top of the real student requests. The search continues with both types of requests. In the $\mathbb{N}_{Ifs}$ phase it is also not permitted to unassign a real student request due to an assignment of a projected student request.

The projected student requests are also weighted using the projected course enrollment sizes. For instance, if there is a course that is expected to have 20 students, there are 10 pre-registered students requesting this course, and there were 16 students enrolled in the course in the previous semester, each projected request of this course is weighted by 0.625 (there are 16 projected requests to cover remaining 10 spaces of the course, therefore 10/16). This means that each such projected request blocks only 0.625 of the limit of each section of the course into which it is assigned.

5 Online Sectioning

After batch sectioning takes place, students can make changes to their schedules using an online interface. During this phase, pre-registered students are allowed to remove themselves from requested courses or to request additional courses and have a new sectioning solution 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. Changes in the course timetable are also possible, potentiality causing some re-sectioning of enrolled students.

New students use the same interface as pre-registered students. They begin by entering an initial set of course requests, based on which they are sectioned to classes in real-time. If they wish to make any changes, they may then continue in the on-line sectioning the same as continuing students (see Fig. 2).



1. ENGL 106
   - ⊞ Lec T 8:30a - 9:20a Full Term HEAV 106
     - ⊞ Lec (a) F 8:30a - 9:20a Full Term HEAV 106
       - ⊟ Lec (b) Th 8:30a - 9:20a Full Term ENAD 130

Sel	*Que*	*Time*	*Date*
◉	☐	Th 8:30a - 9:20a	Full Term

         - ⊟ Rec W 8:30a - 9:20a Full Term HEAV 225

Sel	*Que*	*Time*	*Date*
○	☐	M 8:30a - 9:20a	Full Term
◉	☐	W 8:30a - 9:20a	Full Term

2. BIOL 110
   - ⊞ Lec TTh 2:30p - 3:20p Full Term LILY 1105 K. Mason
   - ⊞ Rec T 6:00p - 6:50p Full Term WTHR 360
   - ⊞ Lab T 3:30p - 5:20p Full Term WTHR 316
   - ⊞ Pso M 4:30p - 5:20p Full Term LILY G126 K. Mason

Fig. 2. 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 student's highest priority course. As it evaluates each possible assignment, the algorithm compares available space with the space expected to be taken by later enrolling 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. Use of historical data to predict distributions in an on-line scheduling application has also been discussed by Bent and Van Hentenryck [3].

5.1 Online Sectioning Algorithm

Online student sectioning is driven by two numbers that are computed for each section from the results of the earlier batch sectioning process. For each course, the section assignments of projected students are analyzed along with all other possible assignments that exist using only those spaces available after all pre-registered students are assigned. Assignments of projected students into other courses are fixed.

Held Space: Space occupied by projected students in the section, i.e.,

$$S_{held} = \sum_{r \in ProjRq(Course(S)), S \in val(r)} w_r,$$

where w_r is a weight of projected student request r for a course with section S, where the projected student is enrolled into the section S. Note that projected student requests are weighted proportionally to the space available in the course (projected size of the course decreased by the number requests from pre-registered students for the course), e.g., if there are 20 spaces available and 10 projected student requests, each request is weighted by 2.

Expected Space: Expresses availability of the section for projected students. Each projected student enrolled in the course containing this section ($Course(S)$ where S is the section) contributes to this number by the portion of the number of available non-conflicting assignments into the course that contain this section among all available non-conflicting assignments into the course, multiplied by the weight of the request.

$$S_{expect} = \sum_{r \in ProjRq(Course(S))} \frac{|\{a | a \in dom(r), conflict(r) = \emptyset, S \in a\}|}{|\{a | a \in dom(r), conflict(r) = \emptyset\}|} w_r$$

For example, if a student can attend either Lecture 1 or Lecture 2 of a course, 1/2 of the request's weight is added to both these lectures.

During online sectioning, students are sectioned one by one as they use the online interface to alter their requirements. Assignments made to students previously sectioned are not changed. A branch & bound technique considering all of a student's requests is used to find the best schedule for the student. The held and expected space counters are used to avoid sections with greater expected demand than available space. These counters are updated after a student is sectioned (i.e., the held space counter is decreased by one for sections the student is enrolled into, and the expected space counter of every section of a course the student is enrolled into is decreased by the proportion of available non-conflicting assignments of the student into the course to all non-conflicting assignments).

The penalty for using a section for a student that is not yet assigned to the course is computed as

$$penalty_{new}(S) = \frac{S_{expect} - avail(S)}{limit(S)},$$

where $avail(S)$ is the available space in the section S and $limit(S)$ is the section limit. For students that are using the online interface to change their assignment, the penalty is

$$penalty_{resect}(S) = \frac{S_{held} - avail(S)}{limit(S)}.$$

It is preferable to use sections that have a space available that is not occupied by students we expect to enroll later in the process. Held and expected space counters are not updated when a student who is already enrolled in the course is re-sectioned (he or she is assigned to a different set of sections than before).

```
procedure ONLINE(student, best, current)
    // parameters: a student and the best and current solution (student's schedule)
    if not better(bound(current),best) then
        return best; // check bound for the current solution
    end if;
    request = nextRequest(student, current); // next request to be assigned
        // in the order of their priority; alternative course requests can be returned
        // only if there is appropriate non-alternative request unassigned in current
    if not request then
        // when current is complete (student will get the requested number of courses)
        // or cannot be extended (e.g., no more requests), save to best and return
        if better(current,best) then best = current;
        return best;
    end if;
    hasEnrl = false; // to be true if there is an available non-conflicting enrollment
    for each enrollment in domain(request) do
        // for each enrollment of the request in the order of their section penalties
        if available(enrollment) and not conflict(current, enrollment) then
            // if all sections are available and not overlapping with current solution
            hasEnrl = true;
            // try to extend the current solution with this enrollment
            best = ONLINE( current ∪ {request/enrollment}, best);
        end if;
    end for;
    if not hasEnrl then
        // there is no available not-conflicting assignment → leave it unassigned
        best = ONLINE( current ∪ {request/∅}, best);
    end if;
    return best;
end procedure
```

Fig. 3. Pseudo-code of the online sectioning algorithm.

The backtracking process (see Figure 3) tries to assign all course requests by a student in their order of priority. For each request it tries various assignments (sets of available sections not conflicting with the higher priority courses) in order based on the number of distance conflicts and the above penalties (the penalty for an assignment is a sum of penalties of its sections). A requested course can be left unassigned only if there is no available assignment not conflicting with one of higher priority. Free time requests are treated in the same way as course requests, except when the solver would be forced to pick a section that is fully reserved ($penalty(S) >= 0$) for students that are expected to come later in the process. This means that when there is an assignment ignoring the free time request using sections that have negative penalties, and all assignments respecting the free time require sections with non-negative penalties, the free time is left unassigned. This rule is quite important in order to prevent students from trying to trick the system into assigning them classes at times that need to be reserved for incoming students. Overall, the process searches for the most complete schedule that minimizes student distance conflicts and section penalties.

When a student is using the online interface to change existing course enrollments, some sections with penalties higher than a given limit may be prohibited. This limit can, for instance, be based on the penalty of the best possible assignment of the student into the course (e.g., a student may be prohibited from receiving sections with a penalty greater than 1.5 times the best possible penalty). It is also possible to disallow use of sections with positive penalties (i.e., sections with more expected demand than available space), especially if the student can be enrolled in the course such that no section with a positive penalty is used.

For a course with only two sections, it is possible to show that the proposed algorithm cannot make a wrong decision when sectioning students with the exact same requests as the projected students that were used to compute the expected space counters. The only case when it can make a bad decision is when a student that can attend both sections is enrolled in a section where all remaining space needs to be available for not-yet-sectioned students that can only attend that section. This would mean that the expected number of students is greater than available space in this section (there is 1 expected student for each yet to be sectioned student than can only attend this section, plus 1/2 for the student being sectioned). However, this means that the other section has fewer expected students than the available space in that section, since the sum of the expected students of these two sections is always equal or less than the total available space in these two sections. The proposed criterion used for selection of a section among sections that are available for that student would, therefore, not take that section and make the bad decision in the first place.

The problem is more complicated for courses with many sections as well as courses that are composed of multiple instructional types (e.g., a lecture and a lab), but empirical experiments show that the proposed approach returns very decent results. See chapter 6.1 for more details.

6 Experiments

In this section experiments using data from Purdue University are presented. These data were gathered from the student registration process for Fall 2007 semester, using a course timetable of over 9,000 sections and 570 rooms. There are 187,847 course requests from 36,117 pre-registered students. Projections based on the Fall 2006 student course enrollments contain 185,494 course requests from 38,740 students. Unfortunately, since the proposed system is still not used in practice, there was no means available to include alternatives, free time requests, or reservations in the test data.

All runs were performed on an Apple Mac Pro server machine with two dual core 3.0 GHz Intel Xeon processors and 4 GB of RAM, using Mac Os X and Java 1.5. The time limit for batch sectioning runs was 8 hours. Branch and bound during online sectioning runs was limited to 1 second per student. On average, the online sectioning algorithm was able to section about 12 students per second.

6.1 Batch versus Online Sectioning

Table 1 compares results achieved using the batch and online sectioning techniques described above. The tests were performed either on all projected students (i.e., student requests from Fall 2006 semester) or on all real (pre-registered) students (i.e., the actual students requests for Fall 2007 semester). All online tests (with both projected and real students) are based on expected and held space counters computed from the batch sectioning run on projected students with the smallest number of unassigned course requests. In this run, there were 316 unassigned course requests and 1,709 student distance conflicts.

Table 1. Batch and online sectioning results. Average and RMS of the number of unassigned requests and student distance conflicts from 10 independent runs are shown.

Problem		Projected Students (Fall 2006)	Real Students (Fall 2007)
Batch Sectioning			
Unassigned course requests (U)		317.40 ± 1.58	264.56 ± 1.88
Distance conflicts (D)		1704.0 ± 15.5	1675.9 ± 17.1
Online Sectioning	U	545.20 ± 13.54	772.45 ± 20.34
Random order	D	1705.5 ± 18.7	1663.5 ± 16.2
Online Sectioning	U	744.00 ± 12.46	1043.70 ± 15.00
Students with more choices first	D	1802.2 ± 15.6	1709.6 ± 11.8
Online Sectioning	U	461.57 ± 3.59	597.00 ± 4.27
Students with fewer choices first	D	1729.8 ± 2.8	1711.3 ± 4.0
Online Section Balancing	U	1538.07 ± 23.78	1481.90 ± 27.39
Random order, no exp/held space	D	1696.4 ± 16.3	1581.3 ± 12.2
Online Section Balancing	U	1753.36 ± 13.85	1721.40 ± 23.44
Students with more choices first	D	1739.8 ± 19.4	1617.7 ± 5.7
Online Section Balancing	U	1288.21 ± 11.07	1249.90 ± 8.37
Students with fewer choices first	D	1747.2 ± 5.8	1663.6 ± 8.7

The unassigned course requests in batch sectioning runs are primarily caused by inconsistencies between student requests and courses available. For projected students, this is the result of changes in courses offered between the Fall 2006 and Fall 2007 schedules. For real students these inconsistencies are mostly caused by capacity limits imposed on courses and sections and the quality of the timetable (i.e., are there adequate non-overlapping time combinations between requested courses to accommodate student demand). To minimize the number unassigned requests due to the timetable, it is very important to consider projected as well as pre-registered students during its construction using an initial sectioning as discussed in Section 3.

Since the schedule of each student in online sectioning depends on the sectioning of previous students (e.g., some sections may be completely filled and no longer available), the overall result may depend greatly on the order in which students are processed. To better understand the effects of ordering, three different scenarios are presented in Table 1. Students are either (1) sectioned in a completely random order, (2) students with more choices are sectioned first, (3) or students with fewer choices are sectioned first. The number of choices is an estimate of how many different schedules are available to each student when all sections are available. The second case represents the worst scenario since students with fewer initial choices are most adversely impacted by a specific class being filled at the time they are sectioned. The result is a smaller number of students able to enroll in all courses required to obtain a complete schedule.

The tests titled *Online Section Balancing* do not use any pre-computed information from a previous batch sectioning run on projected students (i.e., expected and held space counters are zero for all sections at the beginning of the sectioning). Students are processed one by one in the same manner as online sectioning, with expected and held space counters being updated after each student. This helps the process to balance the available space in the sections during the run.

As expected, batch sectioning results in the fewest unassigned requests; however, online sectioning achieves stable and acceptable results as well. The differences when using projected and real students appear to be caused primarily by the discrepancies between projected and real student demand (both cases are based on expected space determined from the batch sectioning of projected students). Having good projections is therefore quite important. It should also be noted that, in practice, only students who did not register before the batch sectioning process, which is run when the university timetable is created and published, will need to be sectioned using the online process. The overall number of unassigned course requests will, therefore, be smaller. This scenario is discussed in more detail in Section 6.3.

Overall, every run of online sectioning performed better than any run of online section balancing (i.e., online sectioning without pre-computed expected space counters). The advantage of using pre-computed information on projected students from batch during online sectioning is clearly visible.

6.2 Student Selection Simulation

In the following experiments, an attempt was made to simulate students having free choice to select sections manually. Since students often have a preference for some times over others, these preferences were modeled based on a set of distributions that correspond to observed behavior. The same algorithm as in online sectioning was used (branch&bound with students taken one by one in a given order), but instead of minimizing sectioning penalties, the algorithm maximizes the preference values on section times for each student. The following distributions were used to create student time preferences:

Mid-Day: For each student, all possible time slots were ordered by drawing them one by one using roulette wheel selection where each time slot has the weight denoted in Table 2. Times before 7:30 am or after 5:30 pm have weight 1.

Table 2. Mid-Day time preferences

	7:30a	8:30a	9:30a	10:30a	11:30a	12:30p	1:30p	2:30p	3:30p	4:30p
Mon	1	4	7	10	10	5	8	8	6	3
Tue	2	4	7	10	10	5	8	8	6	3
Wed	2	4	7	10	10	5	8	8	6	3
Thu	2	4	7	10	10	5	8	8	6	3
Fri	2	4	7	10	10	5	4	3	2	1

This gives times during the middle of the day a higher probability of being preferred than early or late times. The penalty for enrolling a student in a section was computed as the average position of the time slots that a section overlaps with. The overall penalty of all sections in a student's schedule was minimized by use of the branch & bound technique (together with minimization of distance conflicts).

Uniform: The same technique as for *Mid-Day* preferences was used, but all times were drawn with the same weight, i.e.,

$$Weight(time)_{Uniform} = 1.$$

Early/Late: For each student, times were drawn with weights inverse to *Mid-Day* weights, i.e.,

$$Weight(time)_{Early/Late} = 11 - Weight(time)_{Mid-Day}.$$

Figure 4 shows a comparison of the overall number of unassigned course requests for batch sectioning, online sectioning, online section balancing, and the results obtained using student time preferences created from the distributions described above. The *Uniform* distribution returns similar results to online section balancing and, in some sense, it can be seen as a form of section balancing.

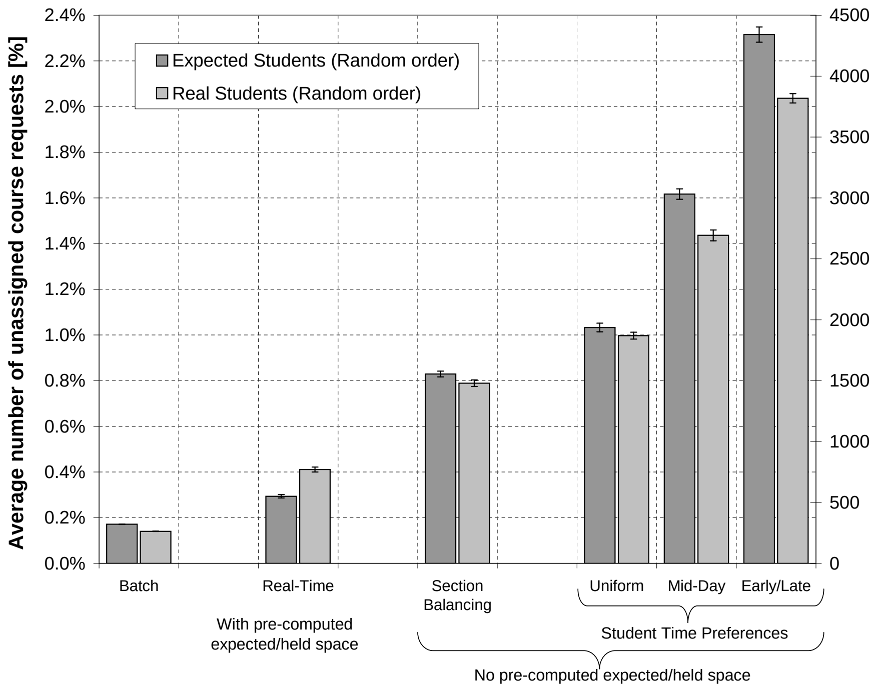


Fig. 4. Comparison of batch sectioning, online sectioning, and student selection simulation on the number of unassigned requests. Average results from 10 independent runs are presented, students are taken in random order in online sectioning.

Mid-day and *Early/Late* returned worse results than online sectioning. This is not surprising since any preference for particular times or sections results in their being filled early and not available to later registering students who may need particular section combinations to obtain a complete schedule with all requested courses at non-overlapping times. The difference between *Mid-day* and *Early/Late* is the result of mismatch in the distribution of sections in the course timetable with student time preferences. The timetable has more classes offered during the middle of the day than on early morning or during evenings due to faculty time preferences.

6.3 Real-world Scenario

In practice not all students are expected to be sectioned using either batch or online sectioning alone. Only students who are already registered will be sectioned using the batch process. Afterward, any additional students would need to be registered using the online sectioning on top of the solution from the batch sectioning phase. This scenario is modeled in the following experiment.

In the results displayed in Figure 5, the actual pre-registered students were split into two groups. The first group of students was sectioned using batch sectioning. (Projected student course requests computed from Fall 2006 student enrollments were included in this process.) The remaining students were then sectioned on top of the batch sectioning solution, using expected space section counters computed from the projected students. In the column labeled *Freshmen* all students in their second or later semester of study were sectioned using batch sectioning and all first semester freshmen students were sectioned using online sectioning. The reason for such a test is that freshmen students are typically not yet on the campus during the Fall course timetabling and batch student sectioning period so they cannot pre-register themselves. The tests labeled *Fm+10%*, *Fm+20%*, ... *Fm+75%*, present scenarios where the given percentage of other (randomly selected) students were also excluded in the batch sectioning process, and therefore had to be sectioned during the online sectioning phase. At Purdue, the number of returning students who do not pre-register for various reasons is expected to be between 20% and 30%. For comparison purposes, cases where all students are sectioned solely using batch or online sectioning are included in Figure 5 in the columns labeled *All Batch* and *All Online* respectively.

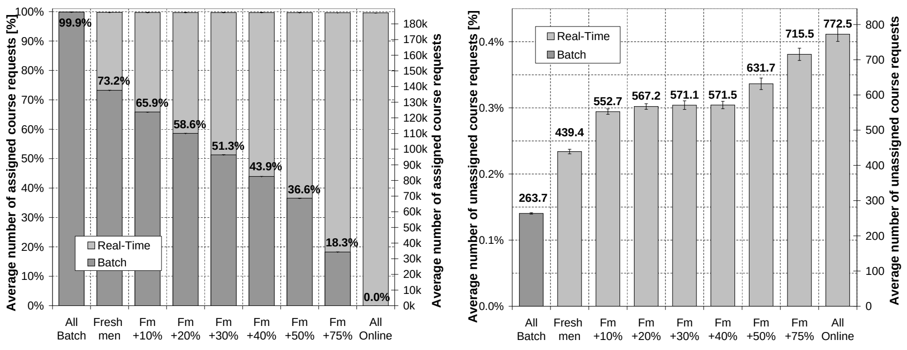


Fig. 5. Combined batch and online sectioning: course requests assigned using batch vs. online (left), unassigned course request detail (right). Average results from 10 independent runs are presented, students were taken in random order in online sectioning.

The results again show the fewest unassigned course requests for batch sectioning, which is to be expected since all requests can be included in the optimization, then monotonically increasing as a greater percentage of students are sectioned by the online process. There appears to be a disproportionately large increase in the number of unassigned students when the first semester freshmen are sectioned online compared to increasing the total number of students sectioned online. The reasons for this are not fully understood, but it appears to be due at least in part to the higher proportion of freshmen in large multi-section courses and the fact that these courses rarely have excess capacity.

6.4 Online Sectioning with Student Choice

Experiments were also conducted to determine the compatibility of the online sectioning approach with providing some degree of choice for students to select different classes. Since the online sectioning algorithm is based on reserving spaces in classes with high expected future demand, the question becomes what the effect is of allowing students to choose among available sections that have an expected demand within some reasonable bounds at the time as he or she is creating or changing a schedule.

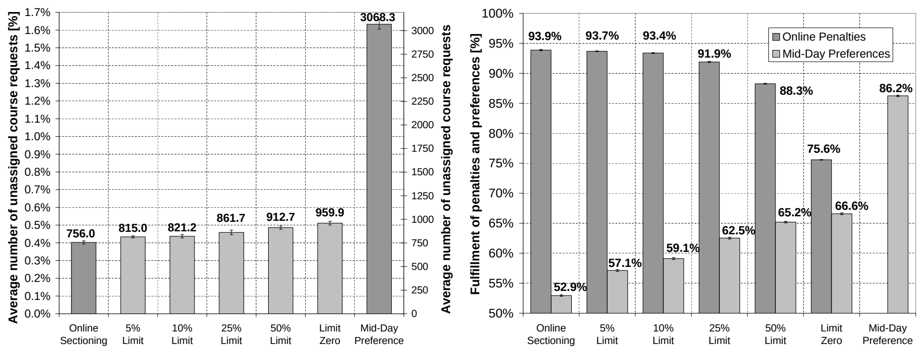


Fig. 6. Online sectioning with student choices: unassigned course requests (left), fulfilment of online penalties versus student preferences of Mid-Day distribution (right). Average results from 10 independent runs are presented, students in random order.

In the results presented in Figure 6, the online sectioning algorithm was altered as follows. For each student, it is first used to find the best solution using the online penalties (expected space counters). Then for each requested course, a limit on this penalty is created based on the online penalty achieved in the computed solution. In the next step, a schedule for the student is computed minimizing the Mid-Day preferences (as in the results presented in Section 6.2), but only sections that have an online penalty within the computed limit are considered. In other words, it attempts to give students some level of choice while still trying to use the computed expectations to divert students from sections with high expected demand. The columns labeled *5% Limit*, *10% Limit*, ... *50% Limit* allow sections with an online penalty not higher than the computed value plus the given percentage. The column labeled *Limit Zero* presents a case when sections with a negative online penalty are always allowed (they have more available space than is expected to be used by incoming students). If an online penalty achieved in the first step is positive, sections with a penalty equal to or below that penalty are allowed. For comparison, a case with all students sectioned using online sectioning (without any choice) and with all students sectioned only based on their preferences (ignoring any expectations) are included on Figure 6, columns *Online Sectioning* and *Mid-Day Preference* respectively. It can be seen that there is an increase in unassigned course requests as the range of acceptable penalty values is increased, indicating increased choice. This increase in unassigned courses is modest, however, compared with sectioning that does not consider the expected demand for each section. Satisfaction of student time preferences, thus degree of choice, also increases as the permitted deviation from the best online penalties values increases.

Figure 7 represents the average number of student choices per course during the online student sectioning process. The horizontal axis shows the number of students that have been sectioned so far. The vertical axis represents the average number of possible course enrollments (section assignment combinations) for the following 100 students. Average values from 20 independent runs are presented.

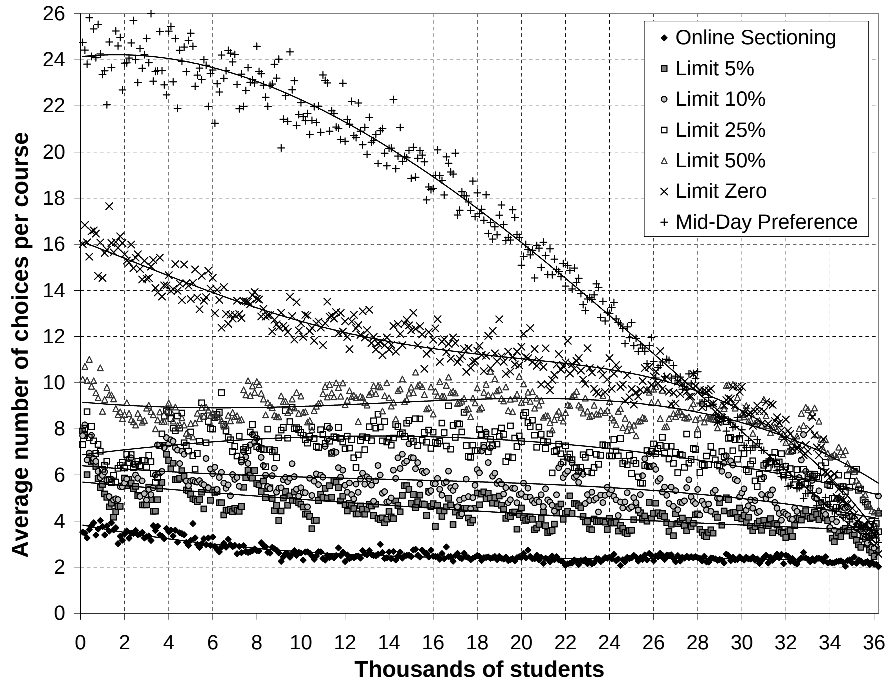


Fig. 7. Online sectioning with student choices: Average number of student choices per course during sectioning, students are taken in random order.

Not surprisingly, the results show that students have the greatest degree of choice when they are free to choose among all of the available sections. The ability to choose rapidly decreases, however, as students fill some of the offered classes. The online sectioning approach that has been proposed, allows for fewer choices of sections until near the end of the registration process, but the number of choices remains more consistent, and therefore more equitable, for all students. Only at the very end, when courses are completely filled, do choices drop off. What is not shown in this graph is the quality of the available choices. As seen from the data in Figure 6, the choices available to students using the proposed online algorithm are better able to fulfill their full set of course requirements. Another very interesting aspects of these results is the effect of allowing the system some flexibility to meet student section preferences. When the limit on the allowable online penalty is raised, the amount of choice also increases as expected, and is maintained through the whole registration process. These results indicate that it is possible to provide students with a reasonable degree of choice and still provide more optimal sectioning solutions with regard to minimizing the number of unassigned course requests. When viewing these results, it should be noted that 36.7% of course requests in the data set are for single section courses and that 59.3% of requests are for courses with four section choices or less. Several very large courses with sections at many times of the day have a great effect on available choices. The presented results were computed with students sectioned in random order, however, similar results were achieved using other orderings.

7 Conclusions

In this paper a practical, real-word problem of student sectioning has been discussed. The three solution phases discussed are designed to accommodate most of the sectioning needs an institution is likely to encounter when automating course timetabling or assigning students to classes based on a fixed timetable. These include the ability to create a course timetable minimizing student time conflicts, to schedule all students to an existing timetable while minimizing conflicts, and to schedule students or make changes to their schedules in real time while anticipating the needs of other students later in the process. The need for some students to include free time requirements or to choose among available sections of a course is also addressed. The experiments discussed clearly show that the proposed approaches are able to improve greatly on an institution's ability to meet student course needs, and offer valuable flexibility to accommodate many student class or time preferences throughout the process.

The student sectioning problem together with the above discussed solution approach are included in a working course timetabling and student sectioning application. This application is publicly available under an Open Source license[1],

[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).

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 and student sectioning problems including the experiments that are discussed in this paper.

References

[1] Mahmood Amintoosi and Javad Haddadnia. Feature selection in a fuzzy student sectioning algorithm. In Edmund Burke and Michael Trick, editors, *Practice And Theory of Automated Timetabling, Selected Revised Papers*, pages 147–160. Springer-Verlag LNCS 3616, 2005.

[2] Don Banks, Peter van Beek, and Amnon Meisels. A heuristic incremental modeling approach to course timetabling. In *Canadian Conference on AI*, pages 16–29, 1998.

[3] Russell Bent and Pascal Van Hentenryck. Online stochastic optimization without distributions. In *ICAPS 2005*. Monterey, CA, 2005.

[4] Vincent A. Busam. An algorithm for class scheduling with section preference. *Communications of the ACM*, 10(9):567–569, 1967.

[5] Michael W. Carter. A comprehensive course timetabling and student scheduling system at the University of Waterloo. In Edmund Burke and Wilhelm Erben, editors, *Practice and Theory of Automated Timetabling III*, pages 64–82. Springer-Verlag LNCS 2079, 2001.

[6] Michael W. Carter and Gilbert Laporte. Recent developments in practical course timetabling. In Edmund Burke and Michael Carter, editors, *Practice and Theory of Automated Timetabling II*, pages 3–19. Springer-Verlag LNCS 1408, 1998.

[7] R. Feldman and M. C. Golumbic. Optimization algorithms for student scheduling via constraint satisfiability. *The Computer Journal*, 33(4):356–364, 1990.

[8] Alain Hertz and Vincent Robert. Constructing a course schedule by solving a series of assignment type problems. *European Journal of Operational Research*, 108(3):585–603, 1998.

[9] Aubin J and J.A. Ferland. A large scale timetabling problem. *Computers and Operations Research*, 16(1):67–77, 1989.

[10] Gilbert Laporte and Sylvain Desroches. The problem of assigning students to courses in a large engineering school. *Computers and Operations Research*, 13(4):387–394, 1986.

[11] Tomáš Müller. Constraint solver library. GNU Lesser General Public License, SourceForge.net. Available at `http://cpsolver.sf.net`.

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

[13] Tomáš Müller, Roman Barták, and Hana Rudová. Conflict-based statistics. In J. Gottlieb, D. Landa Silva, N. Musliu, and E. Soubeiga, editors, *EU/ME Workshop on Design and Evaluation of Advanced Hybrid Meta-Heuristics*. University of Nottingham, 2004.

[14] 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, Selected Revised Papers*, pages 189–209. Springer-Verlag LNCS 3867, 2007.

[15] G. C. W. Sabin and G. K. Winter. The impact of automated timetabling on universities-a case study. *Journal of the Operational Research Society*, 37(7):689–693, 1986.

[16] Scott E. Sampson and Elliott N. Weiss. Increasing service levels in conference and educational scheduling: A heuristic approach. *Management Science*, 41(11):1816–1825, 1995.

[17] Andrea Schaerf. A survey of automated timetabling. *Artificial Intelligence Review*, 13(2):87–127, 1999.

[18] 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.