SchedulExpert: Scheduling Courses in the Cornell University School of Hotel Administration

Timothy R. Hinkin • Gary M. Thompson
School of Hotel Administration, Cornell University, Ithaca, New York 14853
trh2@cornell.edu • gmt1@cornell.edu

This paper was refereed.

A major curriculum review in the School of Hotel Administration at Cornell University revealed that course scheduling was a major problem for the school. We devised a methodology to improve the system and developed a computer program, SchedulExpert, to automate the scheduling process. By using the program, we have eliminated conflicts among core required courses by year and among electives within areas and minimized conflicts among elective sets specified by faculty members. We used to spend weeks on scheduling, but now we achieve better results in only a few hours.
(*Educational systems planning. Computer software. Heuristics.*)

Prior to 1995, the academic administration of the School of Hotel Administration at Cornell University consisted of the dean, the associate dean for academic affairs (ADAA), and the director of graduate studies (DGS). In 1995, the school named a new ADAA and a new DGS and created a new position, the director of undergraduate studies (DUS). As the undergraduate program had doubled between 1990 and 1995, the school felt that the ADAA job had outgrown the ability of any one person to manage it effectively. The school decided that the undergraduate program needed closer oversight and a specific focus and named Tim Hinkin DUS. The school does not have departments but instead comprises seven areas. Students do not select a major but may elect to concentrate in a given area, which requires 12 credits, usually four elective classes that focus on a specific academic discipline.

The first assignment for the DUS was to oversee a major review of the undergraduate curriculum, as it had been over 10 years since one was last conducted. He collected data from industry groups, alumni, faculty, administrators, and students over a 16 month period. After analyzing the data, he learned that, for the most part, the content of courses in the curriculum was appropriate and well suited to industry needs. To the required core, the school added one course, modified the content of several, and rearranged the timing of another. It eliminated several electives and began developing new electives. One of the most important lessons from the review, however, had nothing to do with course content. The review showed that the administration was making it difficult for the students to complete their educational requirements. In addition, faculty members were complaining about having to teach outside of the hotel school's building, where they sometimes encountered inferior classrooms with unreliable technology. Finally, classrooms in the school's own building were often empty. The root cause of these problems boiled down to a single issue: ineffective course scheduling.

Some required core courses were being offered at the same time on the same day at times selected by faculty members who were unaware of any conflicts. This overlap delayed students completing the prerequisites for courses they wanted to take. In one instance, three electives from one area were being offered on the same

0092-2102/02/3206/0045$05.00
1526-551X electronic ISSN

day and at the same time. Students were forced to choose among these electives, all of which they could have used to fulfill their concentration requirements. As a result, students needed two or three terms to get the necessary courses, when they could have taken them all in one term. The university has standard starting times with the majority of courses scheduled to meet Monday, Wednesday, and Friday in 50 minute periods or Tuesday and Thursday in 75 minute periods. Some faculty members had found that this schedule did not fit their pedagogical needs and had determined their own class times. As a result, courses occasionally overlapped by 15 or 20 minutes, forcing students to choose one or the other. Several multiple-hour lab courses were being offered one day per week in the middle of the day, often preventing students from enrolling in other courses that met two or three times per week for a single period. The vast majority of courses were being scheduled between 10:10 AM and 2:40 PM, increasing competition for rooms and forcing many classes out of the building. As a result, many of the classrooms were empty early and late in the day. Hotel school classes often require group projects, and the schedule made it virtually impossible for the students to meet as groups during the day.

The system went through several incarnations.

This apparent chaos did not seem chaotic to an individual faculty member; nor did any single student encounter all of these difficulties. Only when schedules were viewed from a holistic perspective did all of the issues surface. The school offers almost 200 classes, sections, and labs per term, which are taught by 60 faculty members in 21 classrooms of various sizes, shapes, and purposes. Individual faculty members were deciding the days and times to teach and the rooms to use, unaware of the problems they were causing. The situation had come about over the years as faculty members gave their requests to an administrative assistant who neither had the authority to override their instructions nor saw any reason to do so. With scheduling decisions being made at the individual or area level, it was inevitable that some conflicts would exist. It was the magnitude of the problems that was surprising. Hinkin knew that the school was not unique in these practices; in advising his own students, he had noticed that a small department outside the hotel school offered several elective courses at the same time and day. He thought the delivery of education should be viewed as a system and that the school should develop a criteria-based model for scheduling classes. In creating the system, he first had to establish and prioritize scheduling criteria. The number one priority was to eliminate conflicts among required courses. The second was to minimize conflicts among the courses within elective sets that comprise concentrations. The third priority was to maximize building utilization and offer all hotel school courses in its building if possible. The fourth was to give faculty preferred teaching days, and fifth, to give them preferred teaching times. The first change was to standardize class starting times. The second was to pre-enroll incoming freshmen into one of two blocks in which their courses would have no time conflicts. Then, using the criteria, Hinkin manually developed a new schedule for the next two semesters with good results, but it took weeks of work. It was obvious that any changes made in the future might affect other courses, sometimes with unforeseen results, and that a manual system would prove too inefficient in the long run. After one year using the manual system, Tim Hinkin approached Gary Thompson with the problem and the prioritized criteria and asked if technology could provide a better way. He said yes and estimated that he could develop a computer program to solve the problem in two to four weeks, underestimating the time required by about a year.

The State of the Art in Course Scheduling

Reseachers have taken a variety of approaches to course scheduling. These approaches include mathematical programming (Badri 1996, Badri et al. 1998), heuristics (Abramson 1991, Colorni et al. 1998, Cooper and Kingston 1993, Costa 1994, Dowsland 1990, Ferland and Fleurent 1994, Kovačič 1993, Sampson et al. 1995, Stallaert 1997), logic programming (Kang and White 1992), and decision-support systems (DSS) that serve to aid experienced users (Foulds and Johnson 2000,

Johnson 1993). In the case of the hotel school, facilities were a key bottleneck resource and, because of the differences in capacity and equipment in the rooms, any approach that did not explicitly consider room-specific information (such as Badri 1996, Badri et al. 1998, Colorni et al. 1998, Dowsland 1990) would not work.

In its simplest form (Abramson 1991, Kovačič 1993), the course-scheduling, or timetabling, problem has been stated as finding an assignment that fills the timetable (rooms by times) with courses so that certain constraints are satisfied. The hotel school's problem is more complex, because its course lengths vary. Certainly we could define time periods such that no course would ever start or finish except at the starting and ending times of periods. In doing so, however, many of the hotel school's courses would overlap a number of periods. Researchers have considered courses that, at most, cover two periods exactly (Costa 1994, Kang and White 1992). Moreover, we do not assume that faculty are preassigned to courses: our model can determine the most appropriate faculty member to assign to each course.

This is not a particularly desirable role for an untenured professor.

Logic programming, as implemented by Kang and White (1992), offers a viable tool for course-scheduling environments that may permit the scheduler to satisfy all specified constraints. In the hotel school environment, the range and extent of the constraints (Appendix) generally precludes satisfying all constraints (or it would, if many were not formulated as soft constraints). We also see merit in a multicriteria evaluation function, since we feel it is more consistent with the complex nature of the environment. The advantage of a multiple-criteria approach, where users can set the relative priority of the criteria, is that the model does not use a predefined value system. Even at a single institution, administrations, and consequently value systems, change.

Several researchers have formulated the course-scheduling problem as a variant of the quadratic-assignment problem (Mooney et al. 1996, Stallaert 1997). The difficulty we faced with such models was that one of our measures of schedule performance—faculty preference for consecutive classes—could not be satisfactorily measured using only two-variable interaction terms. For example, the hotel school allows faculty members to specify that they prefer to teach their classes consecutively, but do not want to teach more than two (or three) in a row without a break.

Stallaert's (1997) model comes closest to meeting our needs, except that our measure of faculty preference for within-day course timing is more complex than his model allows. Given the size of the hotel school's problem (200 courses, 21 rooms, 25 standard time slots over the week), we deemed a DSS to be insufficient.

We are unaware of any other course-scheduling system, academic or commercial, that has the complexity and flexibility of problem representation that exists in our system (Appendix). Because of the complexity of our representation, we developed an automated course scheduler.

Evolution of the SchedulExpert Class Scheduler

Gary Thompson's development of the scheduling computer program ran well over his initial time estimate because the system went through several incarnations. The program originally started with course information recorded in Excel and a FORTRAN-based dynamic link library (DLL) that read this data, developed the schedule, and passed the results back to Excel. This system worked, but was not robust. At that time, Thompson was untenured and was becoming known as the scheduling tsar, to whom faculty often complained if they did not receive what they saw as an ideal schedule. This is not a particularly desirable role for an untenured professor, yet the lack of robustness of the system at that point would not allow him to detach himself from the process. Eventually, he decided to use an Access database to store the data and a front end for the system written in Microsoft Visual Basic. The scheduling algorithm was also converted to Visual Basic at that time.

He had previously developed the algorithm and written the computer code for scheduling the numerous employees of a large, prominent hospitality firm.

This experience suggested that, for the course scheduler to be useful, it must have several characteristics. First, it should incorporate a rich problem representation, allowing individuals' input and a sense of ownership in the outcome. Second, it must have a scheduling algorithm capable of dealing successfully with this richness. Third, it must have a user-friendly and intuitive graphical interface.

Data Hierarchy

The user interface is very straightforward and simple to use. Data inputs are structured in four levels that build upon each other. The system permits users to easily modify the categories so they can quickly and simply update the system. By users, we mean schedulers using the software both at the hotel school and at other institutions.

Level 1 data requirements are the basic building blocks of the schedule: global parameters, room attributes, scheduling domains, and class start times. Among other things, the global parameters allow the user to specify whether the software should assign faculty members to courses or accept the faculty prespecified for courses, customize abbreviations for days of the week and terms the software uses to refer to courses and faculty, set the timing of three categories of faculty teaching preferences (morning, midday and afternoon), and specify report-formatting options. Room attributes are the exhaustive list of characteristics that rooms collectively possess, such as tiered seating, moveable furniture, Internet hookups, and drafting tables. The user inputs the attribute list and can tailor it to any situation. The user later selects, from the complete attribute list, the characteristics of a particular room, and the characteristics a course requires and the instructor desires in a room. Scheduling domains offer a way of categorizing courses and faculty. For example, a user could define domains at the departmental level, the school level, or the college level to do university-wide scheduling. Class start times are the collective set of all times at which classes can start.

Level 2 data requirements build on the level 1 data inputs: campuses, buildings and rooms, additional start-time data, day-time combinations, and data-collection forms. The user identifies the campuses being scheduled, the buildings on each campus, and the rooms within each building. For each room, the user specifies its attributes: its maximum seating capacity, the minimum allowable course size to schedule in it, the minimum time between courses, any times when it is unavailable, and its desirability index (the software ensures that the more desirable rooms are filled with more and larger courses). The additional start-time data identified at level 2 are whether times are standard (or common) times and the days on which each start time applies. For example, classes that meet Monday, Wednesday, and Friday may start at different times than classes that meet Tuesday and Thursday.

A useful feature is the ability to freeze specific teaching schedules for individual faculty members.

Day-time combinations allow the user to specify how course-offering days match up with start times. Examples of day-time combinations for courses meeting two days per week would be Monday and Wednesday, Tuesday and Thursday, or Wednesday and Friday. The day-time combinations defined at level 2 can be overridden on a course-specific basis at level 3. The data-collection forms are not data inputs per se, but the software does print out forms that facilitate the process of collecting data on courses, faculty, and rooms. The schedulers circulate these forms, which are based on level 1 data, to faculty members as either hard copies or e-mail attachments.

Level 3 data inputs, which build on those at levels 1 and 2, are faculty, courses, campus distances, and building distances. Faculty-specific data include the days and times at which the person is unavailable to teach, days and times they are available but would rather not be teaching, their preferred teaching days and times, their preferences for within-day spacing of courses, ideal number of teaching days, minimum desired time between courses taught on the same day, and the maximum total daily free time between courses. If the software is picking the faculty to teach each course, one also specifies a maximum number of course preps and minimum and maximum weekly teaching hours for each faculty member. One may also specify the courses each faculty member will, or could,

teach. Finally, one specifies a priority value for each faculty member, which indicates the importance the software should place on the schedule requests for that particular faculty member.

Level 3 data specific to each course includes the times at which the course may start (selected from the lists identified at level 1 and 2), the days and times at which the course can or cannot be offered, the attributes required and desired in a room (selected from the list of attributes defined at level 1), the ideal and minimum acceptable room capacity, the ideal building in which to schedule the course, the maximum allowable distance from the ideal building to the building assigned for the course, whether the course must start at the same time in all its meetings, whether the course must meet in the same room for all its meetings, any custom meeting-day patterns for the course, and the list of faculty members preassigned to the course or who could teach the course (in the latter case, also the instructor's skill in teaching the course and preference for teaching it). The software uses the distances (in minutes) between campuses and between buildings on each campus to ensure that instructors have enough time to travel between buildings or campuses between their assigned courses. The distances are also used if a course cannot be assigned to a room within its ideal building.

The highest-level data inputs are the cross-course inputs: courses to disperse, courses listed jointly by several departments, same-day courses, simultaneous courses, and sequential courses. Courses to disperse are those in sets within which there should be no, or minimal, time conflicts. By scheduling the courses in these sets to avoid conflict, the software can develop student-friendly schedules. By prioritizing each group of courses to disperse, the user can ensure that those courses that must not conflict do not conflict. Typically, dispersion groups containing required courses in programs of study are set to preclude conflict, while dispersion groups containing courses in a particular area (or concentration) or courses having high desired coenrollments would be set to minimize conflict because the number of courses in these groups is so great that it may not be possible to avoid conflicts totally.

Joint-listed courses are courses designated by two or more course numbers. Some graduate and undergraduate courses may share the same lecture room, for example. Same-day courses are sets of courses that should meet on the same day(s), such as course recitation sections. Simultaneous courses are those that should be scheduled at the same time. For example, we schedule breakout rooms for a course by creating dummy classes (with lower enrollment) for the breakout "courses" and specify that they should meet at the same time as the original course. Sequential courses are groups of courses that must be scheduled sequentially within a day, sometimes in a defined order.

Scheduler Features

We believe that three features of the software are particularly useful and important: criteria defining a good schedule, pause-and-resume schedule development, and locking in times and rooms.

Thompson structured the software to accomplish three broad scheduling goals: producing a student-friendly schedule, satisfying faculty preferences, and using the facilities effectively. The software determines what classes should be taught at what times and in what locations (and, if desired, who will teach each class). Because scheduling is done at the course level, rather than at the individual student level, the software produces student-friendly schedules by using dispersion groups (sets of classes that should not conflict). In the hotel school, our dispersion groups are each year's core classes, elective classes by area, and cross-area electives that students commonly like to take in the same semester.

Scheduling used to take weeks and now takes a few hours with better results.

We solicit faculty preferences with the data-collection forms. When new people join the faculty, we collect this information and enter it into the database. We assign each faculty member a priority score, with nontenured tenure-track faculty getting the highest priority. Other things being equal, the scheduler will satisfy the preference of higher-priority faculty.

The software manages facility utilization by ensuring that courses are assigned only to rooms that have

desired attributes and a minimum specified seating capacity. The user can set the priority of the different objectives, which also allows evaluation of what-if scenarios.

With such a complex problem and a heuristic solution procedure, we had to decide how long to let the scheduler search for a schedule. The number of solutions generated depends on a number of variables, but in our environment with 200 courses, 60 faculty members, and 21 rooms, the scheduler develops and evaluates approximately 36,000 schedules per hour on a 500 MHz Pentium III personal computer. As the scheduler runs, it displays a performance graph that shows how the best solution yet found compares to earlier schedules on each of the performance criteria. At any time, we can pause the scheduler and examine the characteristics of the best solution. We can save the best schedule, if we like, and then resume the search. We can graphically compare up to four saved schedules at a time on the basis of their performance, which helps us to choose between schedules generated under various what-if scenarios.

Our scheduling process was iterative. We began by developing a schedule and presenting it to the faculty for feedback. This feedback often brought to light missing or incomplete information. If we then redeveloped the schedule, new issues were likely to emerge. Thus, a very useful feature of the software is the ability to freeze, or lock-in, the times (and rooms) of a set of classes and specific teaching schedules for individual faculty members. We then develop a new schedule, changing only the portion of the schedule not locked. This feature has proved useful when we are adding another section of a class or a new elective, for example, since it allows us to find the best time for the new class in relation to the existing schedule. We now have an effective schedule for both fall and spring term locked in. As the school adds courses, we can run the scheduler to find the best times and locations for them based on the input criteria without disrupting the existing schedule.

Implementation Results

The results of the scheduling have been dramatic. We have eliminated conflicts among core required courses by year. By grouping the freshman courses into two blocks that are scheduled without conflicts, we have made it easy for the students to work together in groups outside of class. This past fall we created blocks of courses for the rising sophomore class for the first time. By using the disperse function, we eliminated conflicts among electives within areas and minimized conflicts among elective sets specified by faculty members. We recently graduated the first class of students who spent four years under the new scheduling system. During exit interviews with graduating seniors, we heard almost no complaints about course conflicts and scheduling, and it was the exit interviews five years ago that provided us with much of the information we used to make many of the changes in scheduling.

Several of our purpose-built rooms, such as a drafting room, a hotel computing lab, a meats lab, and a beverage center, continue to be underutilized but are necessary for our particular situation. Our quality flat-floored and tiered classrooms are currently being used at a rate of 87 percent between 8:40 AM and 4:10 PM Monday through Thursday, our primary teaching days. In addition, all hotel school courses are now being taught in the building.

Faculty resistance to the changes was minimal once faculty members fully understood the problems and our objectives. We formally elicited their preferences for days and times and we have been able to satisfy most of their requests. They are all aware that they may not get exactly what they want, but we have been able to get close by seeking their preferences and designing a system that is totally objective and impartial. If someone has a schedule he or she does not like, we can attempt to make changes without affecting the rest of the classes by locking in the majority of courses and scheduling the other courses around them. We can also rotate teaching times among faculty members without creating new conflicts.

Conclusion

Scheduling used to take weeks and now takes a few hours with better results. Through system design and program development, we have made course scheduling more efficient and more effective. Tim Hinkin is training an administrator in the student services office and preparing to hand over the scheduler. He expects

to be consulted only about changes and exceptions. With a locked in schedule and decision criteria and parameters in place, the scheduler now is a tool that can easily be used by the same administrator who had been doing the scheduling prior to 1995.

Based on the successful use of SchedulExpert in the hotel school, Gary Thompson began selling the software on the Internet in March 1999 ⟨http://schedulexpert.com⟩. The software is currently used in schools of law, business, and engineering technology, in departments of math and English, and for scheduling the entire course offerings of small universities and technical schools. The users, whose environments range in size from under 100 to about 800 courses being scheduled per term, collectively span four countries and three continents. Over the past two years, the software has gone through approximately 200 releases, incorporating features based on the suggestions of users and evaluators. Johnson (1993, p. 433) stated his belief that "there is seldom any real likelihood of being able to computerize effectively the task of devising a feasible and acceptable timetable." We believe the SchedulExpert course scheduler, contrary to Johnson's prediction, succeeded because of its rich and realistic problem representation.

APPENDIX

The Course-Scheduling Problem

We present a mathematical model of our course-scheduling problem. A key component of the model is the definition of course-meeting patterns. These patterns, represented by the set P, define the particular meeting-day, start-time, meeting-length, and room-assignment options. In the hotel school, there are over 2,000 unique patterns.

Here is our model:

Indices

c, c'—courses

d—days

f, f'—faculty members

p, p'—meeting patterns (incorporating start times, days offered, meeting lengths, and room options)

r—rooms

Constants

$CrsEnrl_c$ = estimated enrollment of course c.

$CrsIdlCap_c$ = ideal capacity (number of seats) for course c.

$CrsPatDis_{cp}$ = distance between the room assignments of meeting pattern p and the ideal building for course c.

$FacCrsDes_{fc}$ = preference of faculty member f for teaching course c.

$FacCrsSkl_{fc}$ = skill of faculty member f at teaching course c.

$FacIdealDays_f$ = ideal number of teaching days for faculty member f.

$FacMaxDays_f$ = maximum number of teaching days for faculty member f.

$FacMaxWkHrs_f$ = maximum weekly teaching time of faculty member f.

$FacMinDays_f$ = minimum number of teaching days for faculty member f.

$FacMinBtw_f$ = minimum amount of time faculty member f wishes to have between courses.

$FacMinUndes_{fp}$ = number of minutes meeting pattern p would fall in the available but not preferred time of faculty member f.

$FacMinWkHrs_f$ = minimum weekly teaching time of faculty member f.

$FacMaxFree_f$ = desired maximum free time in the daily schedule of faculty member f.

$PatDayStr_{pd}$ = start time of meeting pattern p on day d.

$PatDayFin_{pd}$ = finish time of meeting pattern p on day d.

$PatInRoom_{pr}$ = number of minutes pattern p meets in room r.

$PatMisAttr_{cp}$ = attributes desired by course c that are missing in the room(s) of meeting pattern p.

$PatOnDay_{pd}$ = number of minutes meeting pattern p meets on day d.

$PatRmChng_p$ = number of room changes in meeting pattern p.

$PatStrtVar_p$ = variation in the start times in meeting pattern p.

$PatRmTmDif_{pp'r}$ = number of minutes separating the meeting times of meeting patterns p and p' in room r.

$PatTimDif_{pp'}$ = smallest number of minutes separating the meeting times of meeting pattern p and meeting pattern p'.

$RmCap_r$ = smallest seating capacity of the room assignments in meeting pattern p.

w_i^s = system-controlled weight on objective criterion i (hard constraint).

w_i^u = user-specified weight on objective criterion i (soft constraint).

Sets

C = set of all courses.

$CrsFac_c$ = faculty members sufficiently skilled to teach course c.

$CrsFacDyPt_{cfd}$ = meeting patterns that are consistent with course c and faculty member f and that meet on day d.

$CrsFacPat_{cf}$ = meeting patterns that are consistent with course c and faculty member f.

$CrsPat_c$ = meeting patterns that are consistent with and allowable for course c. The set is selected such that a course can never be assigned to a room with fewer seats than the minimum number the course requires and so that the course is never scheduled in its unavailable time.

$CrsAvoidHrd_c$ = courses that definitely must not conflict with course c.

$CrsAvoidSft_c$ = courses that should not conflict with course c.

$CrsSameDy_c$ = courses that should meet on the same day as course c.

$CrsSimul_c$ = courses that should meet simultaneously with course c.

D = days on which courses may be scheduled.

$DayPatt_d$ = meeting patterns that include day d.

F = faculty.

FacCons = faculty who wish to teach consecutive courses within a day.

$FacCrs_f$ = courses that faculty member f is sufficiently skilled to teach.

FacMinBtwn = faculty who wish a minimum amount of time between their courses.

$FacPat_f$ = meeting patterns that faculty member f is available to teach.

P = meeting patterns for courses (incorporates start times, days offered, and room assignments).

$PatCnflt_p$ = meeting patterns that have time conflicts with meeting pattern p.

$PatTrnst_p$ = meeting patterns that cannot be taught by a faculty member teaching meeting pattern p because of excess transit time.

$PatAsync_p$ = meeting patterns that do not meet simultaneously with meeting pattern p.

$PatDifDy_p$ = meeting patterns that meet on different day(s) than meeting pattern p.

$RmCrs_r$ = courses that can meet in room r.

$RmPat_r$ = meeting patterns that meet in room r.

Variables

x_{cfp} =
$$\begin{cases} 1 & \text{if course } c \text{ is taught by faculty member } f \\ & \text{and assigned to meeting pattern } p, \\ 0 & \text{otherwise.} \end{cases}$$

y_{fd} =
$$\begin{cases} 1 & \text{if faculty member } f \text{ is teaching on day } d, \\ 0 & \text{otherwise.} \end{cases}$$

t_{fd} = total teaching time (in minutes) for faculty member f on day d.

e_{fd} = earliest teaching time for faculty member f on day d.

l_{fd} = latest teaching time for faculty member f on day d.

s_{1f}^- = shortfall in teaching days (below minimum desired) for faculty member f.

s_{2f}^- = shortfall in teaching days (below ideal) for faculty member f.

s_{2f}^+ = surplus teaching days (above ideal) for faculty member f.

s_{3f}^+ = total minutes faculty member f is scheduled in his or her available but not preferred time.

s_{4f}^+ = insufficient minutes between courses for faculty member f.

s_{5f}^- = shortage of teaching minutes per week for faculty member f.

s_{6f}^+ = surplus of teaching minutes per week for faculty member f.

s_{7fd}^+ = surplus of free time in schedule of faculty member f on day d.

s_{8f}^+ = number of consecutive courses taught by faculty member f.

s_9^+ = skill of faculty in their assigned courses.

$s_{10,f}^+$ = preference of faculty member f for his or her assigned courses.

$s_{11,r}^+$ = total enrollment of courses assigned to room r.

$s^-_{12,c}$ = shortage of faculty assigned to course c ($s^-_{12,c}$ = 1 if course c is not scheduled).

$s^-_{13,c}$ = shortage of seats (below the ideal number) of course c in its room assignments.

$s^+_{14,c}$ = number of time conflicts between course c and those courses with which it should not conflict.

$s^+_{15,c}$ = number of courses that should be but are not offered on the same day(s) as course c.

$s^+_{16,c}$ = number of courses that should be but are not offered simultaneously with course c.

$s^+_{17,c}$ = distance from the assigned location of course c and its ideal location.

s^+_{18} = number of room changes within a course across all courses.

s^+_{19} = variation of session start times within a course across all courses.

s^+_{20} = number of room attributes desired by courses that do not exist in the assigned rooms.

Faculty Constraints

Our model contains a number of faculty-based constraints. A necessary requirement of the schedule is that faculty members not be double-booked. Moreover, they need enough time between their classes to travel between buildings or between campuses:

$$\sum_{c\in FacCrs_f}\ \sum_{p\in CrsFacPt_{cf}}\ \sum_{c'\in\{FacCrs_f \mid c'\neq c\}}\ \sum_{p'\in\{CrsFacPat_{c'f}\cap PatTrnst_p\}} x_{cfp}\cdot x_{c'fp'} = 0 \text{ for all } f\in F. \quad (1)$$

The number of days each faculty member spends teaching is controlled in the model. To do this, we must first measure the number of days each faculty member is teaching:

$$y_{fd} \le \sum_{c\in FacCrs_f}\ \sum_{p\in CrsFacDayPat_{cfd}} x_{cfp} \le M\cdot y_{fd} \quad \text{for all } f\in F,\ d\in D. \quad (2)$$

We then determine the number of days each faculty member is scheduled below their minimum specified number:

$$\sum_{d\in D} y_{fd} + s^-_{1f} \ge \text{FacMinDays}_f \text{ for all } f\in F. \quad (3)$$

and limit the maximum number of days each faculty member spends in the classroom:

$$\sum_{d\in D} y_{fd} \le \text{FacMaxDays}_f \text{ for all } f\in F. \quad (4)$$

We also measure the deviation between the number of days each faculty member spends teaching and his or her ideal number of teaching days:

$$\sum_{d\in D} y_{fd} + s^-_{2f} - s^+_{2f} = FacIdealDays_f \quad \text{for all } f\in F. \quad (5)$$

The model attempts to schedule faculty members in their preferred teaching times. To do this, it measures the amount of time each person spends teaching in his or her available but not preferred time:

$$\sum_{c\in FacCrs_f}\ \sum_{p\in CrsFacPat_{cf}} FacMinUndes_{fp}\cdot x_{cfp} - s^+_{3f} = 0 \quad \text{for all } f\in F. \quad (6)$$

Some faculty members want a minimum amount of time between courses, which we measure as follows:

$$\sum_{c\in FacCrs_f}\ \sum_{p\in CrsFacPat_c}\ \sum_{c'\in\{FacCrs_f \mid c'\neq c\}}\ \sum_{p'\in\{CrsFacPat_{c'f} \mid PatTimDif_{pp'}<FacMinBtw_f\}} (FacMinBtw_f - PatTimDif_{pp'})x_{cfp}\cdot x_{c'fp'} - s^+_{4f} = 0 \quad \text{for all } f\in FacMinBtwn. \quad (7)$$

When the software selects the faculty member(s) for courses, it uses each faculty member's minimum and maximum weekly teaching hours, maximum number of course preparations (the number of unique courses taught by a faculty member), and his or her skill at and preference for each of the courses he or she could teach. Here we measure the total daily teaching time:

$$\sum_{c\in FacCrs_f}\ \sum_{p\in CrsFacDyPt_{cfd}} PatOnDay_{pd}\cdot x_{cfp} - t_{fd} = 0 \quad \text{for all } f\in F,\ d\in D. \quad (8)$$

We then measure the shortfall in teaching hours below the minimum weekly level:

$$\sum_{d\in D} t_{fd} + s^-_{5f} \ge FacMinWkHrs_f \quad \text{for all } f\in F, \quad (9)$$

and the surplus of teaching hours about the maximum weekly level:

$$\sum_{d\in D} t_{fd} + s^+_{6f} \le FacMaxWkHrs_f \quad \text{for all } f\in F. \quad (10)$$

To control the amount of free time in each faculty member's daily schedule, we first find the latest time that each person teaches:

$$\sum_{c \in FacCrs_f} \sum_{p \in CrsFacDyPt_{cfd}} PatDayFin_{pd} \cdot x_{cfp} \leq l_{fd} \quad \text{for all } f \in F,\ d \in D, \tag{11}$$

then the earliest time that each person teaches:

$$\sum_{c \in FacCrs_f} \sum_{p \in CrsFacDyPt_{cfd}} PatDayStr_{pd} \cdot x_{cfp} \geq e_{fd} \quad \text{for all } f \in F,\ d \in D. \tag{12}$$

and then we measure the excess free time in each person's daily schedule:

$$l_{fd} - e_{fd} - w_{fd} - s^+_{7f} \leq FacMaxFree_f \quad \text{for all } f \in F,\ d \in D. \tag{13}$$

Some faculty members wish to have their courses scheduled consecutively. We thus measure the extent of consecutive courses in their schedules:

$$\sum_{c \in FacCrs_f} \sum_{p \in CrsFacPt_{cf}} \sum_{c' \in \{FacCrs_f \mid c' \neq c\}} \sum_{p' \in \{CrsFacPat_{c'f} \cap PatCnseq_p\}} x_{cfp} \cdot x_{c'fp'} - s^+_{8f} = 0 \quad \text{for all } f \in FacCons. \tag{14}$$

To match faculty members to courses, we consider each person's skill at teaching the courses:

$$\sum_{f \in F} \sum_{c \in FacCrs_f} \sum_{p \in CrsFacDyPt_{cfd}} FacCrsSkl_{fc} \cdot x_{cfp} - s^+_9 = 0, \tag{15}$$

and preference for the courses he or she can teach:

$$\sum_{c \in FacCrs_f} \sum_{p \in CrsFacDyPt_{cfd}} FacCrsDes_{fc} \cdot x_{cfp} - s^+_{10,f} = 0 \text{ for all } f \in F. \tag{16}$$

Room Constraints

There are only two types of room-based constraints in our model. The first ensures that no room is double-booked (and also that each room has enough time between courses):

$$\sum_{c \in RmCrs_r} \sum_{f \in CrsFac_c} \sum_{p \in CrsFacPat_{cf}} \sum_{c' \in \{RmCrs_r \mid c' \neq c\}} \sum_{f' \in CrsFac_{c'}} \sum_{p' \in \{CrsFacPat_{c'f'} \mid PatRmTimDif_{pp'r} < RmMinBtw_r\}} x_{cfp} \cdot x_{c'fp'} = 0 \text{ for all } r \in R,\ d \in D. \tag{17}$$

We also wish to maximize the utilization of the more desirable rooms. To do this, we first measure the seat utilization of each room:

$$\sum_{c \in RmCrs_f} \sum_{f \in CrsFac_c} \sum_{p \in \{CrsFacPat_{cf} \cap RmPat_r\}} CrsEnr_c \cdot PattMinInRoom_{pr} \cdot x_{cfp} - s^+_{11,r} = 0 \quad \text{for all } r \in R. \tag{18}$$

Course Constraints

Our model includes a number of course-based constraints. In a final schedule, it is desirable that all courses be scheduled. However, users' initial attempts at building a schedule for a term will often not schedule all courses (generally for reasons related to faculty or rooms). The model must be solveable but report the unscheduled courses. Thus, our first course-based measure is whether each course has in fact been scheduled:

$$\sum_{f \in CrsFac_c} \sum_{p \in CrsFacPat_{cf}} x_{cfp} + s^-_{12,c} = 1 \quad \text{for all } c \in C. \tag{19}$$

Courses will never be assigned to rooms that do not have a specified minimum capacity, but each course also has an ideal room capacity. Here we measure the shortfall of seats below the ideal level:

$$\sum_{f \in CrsFac_c} \sum_{p \in CrsFacPat_{cf}} RmCap_p \cdot x_{cfp} + s^-_{13,c} \geq CrsIdlCap_c \quad \text{for all } c \in C. \tag{20}$$

The model must ensure that courses that must not conflict with one another do not conflict (e.g., required program courses):

$$\sum_{f \in CrsFac_c} \sum_{p \in CrsFacPat_{cf}} \sum_{c' \in CrsAvoidHrd_c} \sum_{f' \in CrsFac_{c'}} \sum_{p' \in \{CrsFacPat_{c'f'} \cap PatCnfit_p} x_{cfp} \cdot x_{c'f'p'} \cdot x_{c'f'p'} = 0 \quad \text{for all } c \in C. \tag{21}$$

However, there are also courses for which conflict

should be avoided, if possible (e.g., courses in a major or concentration). Here we measure this type of conflict:

$$\sum_{f\in CrsFac_c}\ \sum_{p\in CrsFacPat_{cf}}\ \sum_{c'\in CrsAvoidSft_c}\ \sum_{f'\in CrsFac_{c'}} \sum_{p'\in\{CrsFacPat_{c'f'}\cap PatCnfit_p\}} x_{cfp}\cdot x_{c'f'p'} - s^+_{14,c} = 0 \quad \text{for all } c\in C. \tag{22}$$

Users sometime desire that some courses meet on the same day as other courses, usually for pedagogical reasons (for example, all the recitation sections for a course should meet on the same day). Here we measure the violations of this constraint:

$$\sum_{f\in CrsFac_c}\ \sum_{p\in CrsFacPat_{cf}}\ \sum_{c'\in CrsSameDyt_c}\ \sum_{f'\in CrsFac_{c'}} \sum_{p'\in\{CrsFacPat_{c'f'}\cap PatDifDy_p\}} x_{cfp}\cdot x_{c'f'p'} - s^+_{15,c} = 0 \quad \text{for all } c\in C. \tag{23}$$

Similarly, one may want a course to meet at the same time that other courses are scheduled:

$$\sum_{f\in CrsFac_c}\ \sum_{p\in CrsFacPat_{cf}}\ \sum_{c'\in CrsSimul_c}\ \sum_{f'\in CrsFac_{c'}} \sum_{p'\in\{CrsFacPat_{c'f'}\cap PatAsync_p\}} x_{cfp}\cdot x_{c'f'p'} - s^+_{16,c} = 0 \quad \text{for all } c\in C. \tag{24}$$

For each course, we assume there will be an ideal building based on who is teaching or who is taking the course. If the course cannot be assigned to its ideal building, we would like to keep it nearby. Here we measure the distance between the building to which the course is assigned and the ideal building for the course:

$$\sum_{c\in C}\ \sum_{f\in CrsFac_c}\ \sum_{p\in CrsFacPat_{cf}} CrsPatDis_{cp}\cdot x_{cfp} - s^+_{17,c} = 0. \tag{25}$$

For courses that we allow to meet in different rooms on different days, we may still wish to keep the room changes to a minimum. Consequently, we measure the extent of room changes:

$$\sum_{c\in C}\ \sum_{f\in CrsFac_c}\ \sum_{p\in CrsFacPat_{cf}} PatRmChng_p\cdot x_{cfp} - s^+_{18} = 0. \tag{26}$$

Similarly, we would like to keep the variation in course start times to a minimum for those courses that we allow to start at different times on different days. We measure the start-time variation as follows:

$$\sum_{c\in C}\ \sum_{f\in CrsFac_c}\ \sum_{p\in CrsFacPat_{cf}} PatStrtVar_p\cdot x_{cfp} - s^+_{19} = 0. \tag{27}$$

Finally, although a course is never assigned to a room that lacks certain required attributes, the course could be assigned to a room that lacks some desired attributes. We measure the number of missing attributes as follows:

$$\sum_{c\in C}\ \sum_{f\in CrsFac_c}\ \sum_{p\in CrsFacPat_{cf}} PatMisAttr_{cp}\cdot x_{cfp} - s^+_{20} = 0. \tag{28}$$

Objective

The objective includes components for which the user can set the relative priorities, as well as components for which the system sets the weights. The system-controlled weights are placed on those criteria measuring violations of hard constraints. We describe each component below.

$$\begin{aligned}\text{MIN } Z = {} & \sum_{f\in F} FacWt_f(w^u_1\cdot(s^-_{2f}+s^+_{2f}) + w^u_2\cdot s^-_{3f} - w^u_5\cdot s^+_{10,f}) + \sum_{f\in FacMinBtwn} FacWt_f(w^u_3\cdot s^+_{4f}) \\ & - \sum_{f\in FacMinBtwn} \mathrm{FacWt}_f(w^u_4\cdot s^+_{8f}) + \sum_{f\in F}(w^u_6\cdot s^+_{9f}) + w^u_7 \sum_{r\in R} RmDes_r\cdot s^+_{11,r} + \sum_{c\in C}(w^u_8\cdot s^-_{13,c} + w^u_9\cdot s^+_{14,c} + w^u_{10} \\ & \cdot CrsEnrl_c\cdot s^+_{17,c} + w^u_{11}\cdot s^+_{18,c} + w^u_{12}\cdot s^+_{19,c}) + \sum_{f\in F}(w^s_1\cdot s^-_{1f} + w^s_2\cdot s^-_{5f} + w^s_3\cdot s^+_{6f} + w^s_4\cdot s^+_{7f}) \\ & + \sum_{c\in C}(w^s_5\cdot s^-_{12,c} + w^s_6\cdot s^+_{15,c} + w^s_7\cdot s^+_{16,c}) + w^s_8\cdot s^+_{20}.\end{aligned} \tag{29}$$

The criteria controllable by the user can be categorized as faculty based, room based and course based. Each of the faculty-based performance measures is weighted by the user-defined importance of the faculty members. The faculty performance measures controlled by the user are, in order of appearance in (29),

minimizing deviations from the ideal number of teaching days; minimizing the time scheduled in the available but not preferred teaching times; maximizing preference for their assigned courses; minimizing insufficient desired time between courses; maximizing the number of consecutive courses; and maximizing the skill faculty members have for teaching their assigned courses. The room-based criterion is maximizing the utilization of the desirable rooms. Course-based performance measures under user control are minimizing the shortfall of seats below the ideal level; minimizing conflicts between courses that should not conflict; minimizing the students' minutes of distance to travel from the ideal buildings; minimizing room changes; and minimizing variation in start times.

The criteria with importance controlled by the system are faculty based and course based. The faculty-based criteria, in order of appearance in (29), are minimizing the number of teaching days below the minimum acceptable; minimizing shortfalls and surpluses in teaching hours; and minimizing excess daily free time. Course-based criteria are minimizing the number of unassigned courses; minimizing instances in which courses that should be scheduled on the same day are not; minimizing instances in which courses that should be scheduled simultaneously are not; and minimizing the number of missing desirable room attributes.

Solving the Model

The complexity of the scheduling problem necessitated a heuristic solution procedure. Gary Thompson developed a customized simulated annealing-based heuristic for the problem. The heuristic first constructs a schedule from scratch. This schedule is both the incumbent and the best. The heuristic then iteratively perturbs the incumbent schedule to arrive at a trial schedule and uses standard simulated annealing logic to determine whether to replace the incumbent schedule with the trial. The replacement always occurs if the trial schedule is better than the incumbent. An inferior trial schedule can also replace a superior incumbent schedule, which allows the algorithm to backtrack. The probability of accepting an inferior schedule declines with higher degrees of inferiority and over time as the algorithm executes.

The construction process begins by randomly selecting an unscheduled course and then evaluating 100 randomly selected possibilities for its scheduling. If any of the alternatives will work for the course, the algorithm randomly selects the actual assignment from the best three possibilities. The construction process terminates when all courses have been scheduled or when the remaining courses cannot be scheduled.

The algorithm perturbs the schedule in the following way: it drops a selected number of courses from the schedule (that is, it sets their status to unassigned). The number dropped ranges from one course to three percent of the courses in the schedule. The algorithm then uses the construction phase to schedule, if possible, the currently unscheduled courses.

The algorithm includes an automatic restart feature if it develops a specified number of schedules without finding a new best solution. This feature is very useful when users run the algorithm overnight (as we typically recommend for a production run), since it ensures that the algorithm will not simply report the best schedule it found early in the run. Typically, users of the software will do a preproduction run of about five minutes to see if any data issues may be precluding all courses from being scheduled. Our experience has been that users are comfortable running the scheduler overnight, since the users perform scheduling only a few times a year, and they increase their likelihood of finding a very good schedule with a longer run.

In developing the algorithm, we opted for a modularized structure that makes it easy to add new criteria or constraints, rather than a very specialized structure oriented towards the speed of the algorithm. Our experience has been that users want realistic schedules: a fast algorithm is of little value if it does not generate schedules that meet users' relevance criteria.

References

Abramson, D. 1991. Constructing school timetables using simulated annealing: Sequential and parallel algorithms. *Management Sci.* **37**(1) 98–113.

Badri, M. A. 1996. A two-stage multi-objective scheduling model for faculty-course-time assignments. *Eur. J. Oper. Res.* **94**(1) 16–28.

——, D. L. Davis, D. F. Davis, J. Holllingsworth. 1998. A multi-objective course scheduling model: Combining faculty preferences for courses and times. *Comput. Oper. Res.* **25**(4) 303–316.

Colorni, A., M. Dorigo, V. Maniezzo. 1998. Metaheuristics for high school timetabling. *Comput. Optim. Appl.* **9**(3) 275–298.

Cooper, T. B., J. H. Kingston. 1993. The solution of real instances of the timetabling problem. *Comput. J.* **36**(7) 645–653.

Costa, D. 1994. A tabu search algorithm for computing an operational timetable. *Eur. J. Oper. Res.* **76**(1) 99–110.

Dowsland, K. A. 1990. A timetabling problem in which classes are inevitable. *J. Oper. Res. Soc.* **41**(10) 907–918.

Ferland, J. A., C. Fleurent. 1994. SAPHIR: A decision-support system for course scheduling. *Interfaces* **24**(2) 105–115.

Foulds, L. R., D. G. Johnson. 2000. SlotManager: A microcomputer-based decision support system for university timetabling. *Decision Support Systems* **27**(4) 367–381.

Johnson, D. 1993. A database approach to course timetabling. *J. Oper. Res. Soc.* **44**(5) 425–433.

Kang, L., G. M. White. 1992. A logic approach to the resolution of constraints in timetabling. *Eur. J. Oper. Res.* **61**(3) 306–317.

Kovačič, M. 1993. Timetable construction with Markovian neural network. *Eur. J. Oper. Res.* **69**(1) 92–96.

Mooney, E. L., R. L. Rardin, W. J. Parmenter. 1996. Large-scale classroom scheduling. *IIE Trans.* **28**(5) 369–378.

Sampson, S. E., J. R. Freeland, E. N. Weiss. 1995. Class scheduling to maximize participant satisfaction. *Interfaces* **25**(3) 30–41.

Stallaert, J. 1997. Automated timetabling improves course scheduling at UCLA. *Interfaces* **27**(4) 67–81.

Judi Brownell, Associate Dean for Academic Affairs, 146A Statler Hall, Cornell University, Ithaca, New York, writes: "In the mid-1990s we had many problems with course conflicts and classes that were taught outside of our building. With the development of the scheduling program we have virtually eliminated all conflicts and kept all of our courses in the building where faculty have adequate computer access and audio/visual equipment. Any subsequent changes to the schedule can be made without worrying about creating new problems. It took several years and iterations, but we now have a scheduling process that meets the needs of students, faculty, and administration."