Abstract

The problem of producing a schedule is a very demanding task for any academic institution. To deal with this complex large scale timetabling problem, we propose a decision support system in which efficient optimization procedures are available. The user can then operate with both sophisticated DSS features as well with specialized methods that can help him with the combinatorial aspect of the problem.

Keywords : Schedule, Timetabling, Decision Support System, Heuristic Procedures, Recursivity.

Producing course schedules for an academic institution is a difficult task since we have to account for student registrations, lecturers and classroom availabilities, as well as other constraining factors. The corresponding integer mathematical programming problem is difficult to deal with. Furthermore, most practical applications give rise to large scale problems. Consequently, heuristic methods have been proposed to treat the problem. Several techniques have been adapted and will be mentioned in the next section.

However, since many parties are involved in the problem (students, teachers, school administrators), another approach has been taken to handle the problem. Decision Support Systems (DSS) were developed in order to satisfy some of the constraints that are hard to formulate mathematically [Kassicieh et al. 1986; Glassey and Mizrach 1986].

We believe that a DSS can really be useful when powerful computerized optimization procedures are offered to the user, as well as other sophisticated DSS features. This paper describes such a user-friendly DSS that is implemented on a PC-type microcomputer and requires no previous knowledge of computers on the part of the user.

Problem

Producing a course schedule is a complex problem involving many factors. It consists of two subproblems: a timetabling problem specifying the master timetable, and a grouping problem specifying groups of students for large courses having multiple sections. Conflicts for students, lecturers and rooms have to be avoided, and constraints on lecturers and room availabilities have to be respected. (Note that there is a conflict whenever two lectures taking place simultaneously require the same room, or have common students or lecturers.) Many of the approaches for the timetabling problem are summarized in de Gans [1981] and Schmidt and Strohlein [1980]. Bloomfield and McSharry [1979], Knauer[1974], Almond [1969], and Selim [1983] propose heuristic procedures. Mulvey [1982] suggests a network model, and Tripathy [1980,1984] uses Lagrangian relaxation to solve the timetabling problem. Carlson and Nemhauser [1966], Davis et al. [1974], and Ferland and Roy [1985] formulate it as a problem of assigning conflicting activities to resources. On the other hand, the grouping problem has been analyzed by Colijn [1973], Winters [1971], and Laporte and Desroches [1986].

Since the two components of the problem are closely related, a complete solution approach should be able to treat them simultaneously. More recently, Aubin and Ferland [1989] and Hertz [1991] used such an approach, and proposed a model including two sets of decision variables. This model can be found in the appendix.

The usual approach to dealing with the problem was to first generate a timetable and then load the students into it to obtain the grouping. Aubin and Ferland [1989] proposed an iterative procedure in which both the timetable and the grouping are modified at each iteration until no improvement is possible. In our system, described below, several methods are available to deal with the timetabling and grouping subproblems at different stages of the process. These algorithms generate good results with much better solutions than those obtained manually. However, these solutions will probably have to be modified in order to account for some constraints which are difficult to express mathematically. The best schedule is not always the one with the least conflicts. For instance, special care may have to be taken when considering certain lecturers or specific courses, or for students enrolling in a particular program. Furthermore, last minute changes may imply local modifications to the schedule. In such a case, the user not only needs interactive solution editing tools, but also powerful interactive optimization tools capable of providing appropriate alternatives, according to his specifications. Such tools are also embedded into our DSS.

The system

The system consists of a set of pop-up menus in which functions are accessed easily via a moving cursor. These functions can be divided into 5 categories:

Data handling: Appropriate tools are available to input and modify the data. Some of these data can easily be imported from existing files of other systems. Data on courses, lecturers, rooms, and students are required. For each course, the following information must be provided: the lecturers teaching, the number of lectures, their length, a preliminary schedule (first period being on Monday morning, if not otherwise specified), and the type of room required. The user can preassign times or specific rooms to lectures. In such cases, the computerized optimization procedures will not modify these assignments. For each lecturer, his availabilities and preference for each period are to be provided. A set of available rooms also has to be specified. This set is partitioned into different types according to their characteristics, e.g. their capacity. Some specialized rooms are unique, e.g. physics laboratory, and they are identified as such. Finally, for each student, we need the list of courses in which he is registered. Alternatively, if student registrations are not available, registrations of typical students following different program streams can be used. This information is required to generate the conflict matrix.

If the user wishes to generate several solutions, each of these can be stored on disk and reloaded upon request.

Automatic optimization: Several heuristic procedures are available to deal with the problem as a whole or with some specific subproblems. These procedures depend on parameters that can be set by the user.

Interactive optimization: Tools are available to provide decision-making support whenever the user has to modify a solution. This support can range from displaying information to the user in order to guide his efforts, to suggesting alternative schedulings of a lecture according to the user's requirements.

Reports generator: Several reports on data and solutions can be generated, and then stored, displayed or printed.

Parameters setting: Parameters for automatic and interactive optimization can be easily fixed or modified by the user.

Each function executes a particular task and there is no specific order in which most of these are to be called. The system always works on a current solution that can be modified with editing tools or by several applications of automatic or interactive optimization procedures with different values of the parameters. Intermediate solutions can easily be stored on disk, and later, any of these can be restored as the current solution. Figure 1 shows the interaction between different tools embedded in the DSS.

A typical sequence of actions using the different functions of the system to deal with a specific course scheduling problem can be the following:

i) All data are input to the system using the data handling tools.

S_i denotes an intermediate solution on disk, and S the current solution.

Figure 1: DSS tools interaction

ii) The user can make some commitments by fixing the schedule or the room for some lectures.

iii) The user specifies the parameters, or their values are fixed by default.

iv) The automatic procedures are then applied to the data. The user selects the heuristic procedure to deal with the timetabling subproblem.

v) The generated solution can be evaluated with regard to conflicts, classroom shortage and assignment cost. If the solution is too far from being acceptable, then, in order to improve the solution, the user may wish to reapply the automatic procedures with different parameter values, or make alternate or additional commitments, or use alternate procedures to deal with the timetabling subproblem. For instance, if there is a large number of lecturer conflicts, the user can increase the corresponding weight in the objective function or try to change some lecturer availabilities, if possible.

vi) If the current solution is close to being acceptable, then it can be made fully acceptable using the interactive optimization tools.

vii) Finally, appropriate reports can be generated.

Since the system saves the intermediate solutions, any of these can be selected as the current solution at any stage of the solution process.

Furthermore, the system can be useful during the term to modify the course schedule if some data change, e.g. lecturer or room availability. Interactive tools or specific automatic procedures can be used to update the solution as required. Of course, appropriate commitments fixing the schedule or room for most lectures are required to reduce the impact of the modifications.

Finally, the system can be used to deal with students registering during the term or modifying their registration. Indeed, the student grouping procedure applied for these students will generate their schedule.

Automatic optimization

The automatic components of the system can be used to generate an initial solution to the problem which the user may then modify as desired. These components are based on heuristic algorithmic methods for solving the mathematical models associated with the different timetabling and grouping subproblems introduced in the appendix. A more detailed description of these components (models and methods) can be found in [Aubin and Ferland 1989] or [Hertz 1991]. Automatic methods are available to execute the following tasks:

- Timetabling
- Student grouping
- Classroom assignment

The timetabling subproblem is often more difficult to solve than the other two. Two different methods for solving this subproblem are included in the system, and they are both applied for fixed values of the grouping variables.

i) Reassignments method [Aubin and Ferland 1990]. Given a current timetable, all the teaching hours of the week are scanned. If some lectures generate conflicts or classroom shortage, exchange procedures are used to modify the schedule of some of these in order to reduce the objective function specified in the model of the appendix. The values of some parameters are given by the user to specify the relative weights of the function components (student and lecturer conflicts, classroom shortages, lecturer availabilities).

ii) Constructive method. Lectures are selected at random and are scheduled sequentially, explicitly satisfying the constraints as specified by the user rather than through costs as in the model of the appendix. The constraints prohibit any student, lecturer or classroom conflict and ensure the satisfaction of lecturers and the availability of classrooms. Other additional constraints can be included such as, for example, a constraint prohibiting two lectures of the same course to be scheduled on the same day. Some of these constraints can be relaxed through parameter setting. For example, the student conflict constraint is violated if more than N students have a conflict in their schedules, where N is a parameter. The more we increase N from 0, the more we relax this constraint.

Some lectures that are already scheduled may have to be moved in order to schedule a specific lecture. This rescheduling process is done through a scheduling operator using a procedure that may have to be called recursively for rescheduled lectures. The recursivity level limiting the number of lectures to be rescheduled is also a parameter fixed by the user. This scheduling operator is described in the appendix.

The student grouping procedure [Aubin and Ferland 1990] is a straightforward branch and bound partially enumerative method that is applied for each student. The classroom assignment is done at the end of the process when an acceptable timetable has been identified by the user. It is completed according to a straightforward assignment procedure. Indeed, since we take into account the number of available rooms at each period during the timetabling subproblem, the specific rooms are assigned directly to each lecture. We only have to check that once a classroom is assigned to a lecture, it is kept for its entire length.

These methods can be used individually according to a specific solution strategy of the user. The system also provides a default solution strategy in which one of the timetabling and the grouping procedures are called successively until no improvement is possible. Then, the classroom assignment procedure is applied.

Interactive optimization

Two different tools are available to assist the user in modifying solutions that are generated with the automatic optimization procedures. They both allow the rescheduling of lectures one at a time.

With the first tool, the user first identifies the lecture to be rescheduled. The system then displays on the screen a partial schedule including all lectures having students or lecturers in common with the identified lecture. For each period, the number of students in conflict (and their identification, upon request), the number of available rooms or their shortage, and the lecturer's availability are given. The user can easily navigate through this information and scroll up or down to examine each period.

Using his own judgement with the aid of the aforementioned tool, the user can decide at which time he will reschedule the lecture. However, it may sometimes be useful to have an even more powerful procedure to examine more complex modifications of the timetable. An interactive optimization technique allows the examination of several potential schedulings of a lecture involving rescheduling of others. The scheduling operator described in the appendix is used for this. For instance, scheduling lecture l_1 at period p_1 may require rescheduling lecture l_2 from period p_1 to period p_2 and rescheduling lecture l_3 from period p_2 to period p_3. Hence, for a specific lecture, the system can propose several alternate ways to modify the schedule, and the user selects the one that he prefers. A similar procedure was also used successfully for sport scheduling and is described in detail in [Ferland and Fleurent 1987,1990].

It is interesting to note that the scheduling operator is used both in the constructive method and in interactive mode. However, with automatic optimization, the first possible lecture schedule alternative is selected. In the interactive mode, the user selects the scheduling he prefers. Hence, the selection can be made according to some additional constraints which may be hard to formulate mathematically. Finally, if some constraints are relaxed or the recursivity level is increased, then the scheduling operator will propose more alternatives to the user.

Results and implementation

The system is implemented in Turbo-Pascal on a PC-type micro-computer using DOS. Because of memory space limitations, special structures are used for some of the data. For instance, sparse matrix storing techniques [Eisenstat et al. 1981] are used to store the conflict matrix. In general, less than 4% of the elements of these matrices are non-zero. For very large problems, only boolean bits are stored during some phases of the algorithms (automatic and interactive procedures for the timetabling subproblem) to indicate if there exists a conflicting situation between two courses.

We conducted tests in three different institutions: a university department (P_1), a university faculty (P_2), and a large high school (P_3). Table 1 shows how much time is necessary for the automatic procedures to obtain good solutions. The quality of the solution may depend upon the user's choices and behavior. However, the values of the ratio RC

$$RC = \frac{\text{\# hours of student conflicts}}{\text{\# of hours of lectures attended by students}}$$

(evaluated after using only the automatic procedures) indicate the quality of the solutions as far as students conflicts are concerned.

To fully appreciate the quality of these solutions, we have to be aware that in cases (P_1) and (P_2), the student registrations were not constrained by an existing timetable and that the schedule is constructed by the system according to the students' choices and the other constraining factors. In other words, the schedule adapts itself to the students' choices rather than the students adapting to the schedule.

In case (P_1), the department has been experimenting with the system and earlier versions since early 1980. The constraints on classrooms are not included in this case (i.e. $\rho = 0$), and student registrations are partly actual (undergraduate students) and partly typical (graduate students). The person in charge can now use the system with minimal support from us.

The university faculty (P_2) has been using the system (and earlier versions of it) for the last three years. In this case, actual student registrations are used, and classroom shortages is a crucial issue. Furthermore, since the faculty calls on several part-time lecturers with restrained availabilities to teach, this adds to the difficulty of generating an acceptable schedule.

Finally, case (P_3) was experimental to evaluate how the system can be adapted to deal with problems larger than (P_2). Here, the high school is planning to schedule using typical student registrations. Taking into account that the schedule has to be prepared only once every semester, and taking into account the effort to prepare

Problem	# of students	# of course sections	# of lectures	RC	CPU time (80386 processor)
P_1	500	90	200	.00286	4.5 minutes
P_2	1500	160	350	.00280	17 minutes
P_3	6000	2000	3000	.00261	6.4 hours

Table 1: Results

this schedule manually, the experiment shows the capability of the system to deal efficiently with such a large problem. Therefore, our DSS should soon be able to help institutions as large as (P_3).

Conclusion

There are two major difficulties with course scheduling. First, the complexity of the corresponding mathematical problem renders the solution of it beyond human capability. Second, a lot of factors make active human intervention essential during the solution process. For such an application, a DSS should therefore include both powerful OR procedures to tackle the problem, and easy access to interactive tools to control the solution process. With our application, the two aspects proved to be closely complementary, and the lack of either of these would result in a big reduction in the quality of the final solution.

Acknowledgements

The authors would like to express their sincere gratitude to the following practitioners for discussing their problems and approaches: Lucille Roy from the Computer

Science and Operations Research department of the University of Montreal, Jean Goulet and Gaston Lacroix from the Science Faculty of the Sherbrooke University, Pierre Beauchamp and Sam Kay from Vanier College in Montreal.

References

[1] **Almond, D.M.,** "A University Faculty Timetable", *The Computer Journal* 12 (1969) 215-217.

[2] **Aubin, J., Ferland, J.A.,** "A Large Scale Timetabling Problem", *Computers and Operations Research* 16 (1989) 67-77.

[3] **Bloomfield, S.D., McSharry, M.M.** "Preferential Course Scheduling", *Interface* 9 (1979) 24-31.

[4] **Carlson, R.C., Nemhauser, G.L.,** "Scheduling to Minimize Interaction Cost", *Operations Research* 14 (1966) 52-58.

[5] **Carter, M.W.,** "A Survey of practical applications on examination timetabling", *Operations Research* 34 (1986) 193-202.

[6] **Colijn, A.,** "A Sectioning Algorithm", *INFOR* 11 (1973) 210-225.

[7] **Davis, F.E., Devine, M.D., Lutz, R.P.,** "Scheduling Activities Among Conflicting Facilities to Minimize Conflict Cost", *Mathematical Programming* 6 (1974) 224-228.

[8] **Eisenstat, S.C., Schultz, M.H., Sherman A.H.,** "Algorithms and Data Structures for Sparse Symmetric Gaussian Elimination", *SIAM Journal on Scientific and Statistical Computing* 2 (1981) 225-237.

[9] **Ferland, J.A. and Roy, S.,** "Timetabling Problem for University as Assignment of Activities to Resources", *Computers and Operations Research* 12 (1985) 207-218.

[10] **Ferland, J.A., Fleurent C.,** "Computer Scheduling for a Sport League", *INFOR* 29 (1991) 14-25.

[11] **Fleurent, C.** "Méthodes Heuristiques Pour la Conception de Calendriers Sportifs", Master Thesis, Département d'informatique et de recherche opérationelle, Université de Montréal (1987).

[12] **de Gans, O.B.,** "A Computer Timetabling System for Secondary Schools in the Netherlands", *EJOR* 7 (1981) 175-182.

[13] **Glassey, C.R., Mizrach, M.,** "A Decision Support System for Assigning Classes to Rooms", *Interfaces* 16 (1986) 92-100.

[14] **Hertz, A.,** "Tabu Search for Large Scale Timetabling Problems", *EJOR* 54 (1991) 39-47.

[15] **Kassicieh, S.K., Burleson, D.K., Lievano, R.J.,** "Design and Implementation of a Decision Support System for Academic Scheduling", *Information and Management* 11 (1986) 57-64.

[16] **Knauer, B.A.,** "Solution of Timetable Problem", *Computer and Operations Research* 1 (1974) 363-375.

[17] **Laporte, G., Desroches, S.** "The Problem of Assigning Students to Course Section in a Large Engineering School", *Computer and Operations Research* 13 (1986) 387-394.

[18] **Mulvey, J.M.,** "A Classroom/Time Assignment Model", *EJOR* 9 (1982) 64-70.

[19] **Schmidt, G., Strohlein, T.** "Timetabling Construction-An Annotated Bibliography", *The Computer Journal* 23 (1980) 307-316. 1473-1489.

[20] **Selim, S.M.,** "An Algorithm for Producing Course and Lecturer Timetables", *Computers and Education* 7 (1983) 101-107. 1473-1489.

[21] **Tripathy, A.,** "A Lagrangian Relaxation Approach to Course Timetabling", *Journal of Operational Research Society* 31 (1980) 599-603.

[22] **Tripathy, A.,** "School Timetabling-A Case in Large Binary Integer Linear Programming", *Management Science* 30 (1984) 1473-1489.

[23] **Winters, W.K.,** "A Scheduling Algorithm for a Computer Assisted Registration System", *Communications of the ACM* 14 (1971) 166-171.

Appendix

Model

i) timetabling variables: for lecture i and starting time j, $1 \leq i \leq n, 1 \leq j \leq m$,

$$x_{ij} = \begin{cases} 1 & \text{if lecture } i \text{ starts at time } j \\ 0 & \text{otherwise;} \end{cases}$$

ii) grouping variables: for student s and course section a, $1 \leq s \leq S, 1 \leq a \leq A$,

$$y_{sa} = \begin{cases} 1 & \text{if student } s \text{ is assigned to course section } a \\ 0 & \text{otherwise;} \end{cases}$$

The model is then specified as follows:

$$\text{Min} f(x,y) = f_1(x) + \mu f_2(x,y)$$

subject to:

$$\sum_{j=1}^{m} x_{ij} = 1 \qquad 1 \leq i \leq n \tag{2.1}$$

$$x_{ij} = 0 \text{ or } 1 \qquad 1 \leq i \leq n, i \leq j \leq m \tag{2.2}$$

$$\sum_{a \in Q_w} y_{sa} = 1 \qquad 1 \leq s \leq S, w \in R_s \tag{2.3}$$

$$y_{sa} = 0 \text{ or } 1 \qquad 1 \leq s \leq S, i \leq a \leq A \tag{2.4}$$

where R_s denotes the courses in which student s is registered and Q_w is the set of course sections of course w. Constraints (2.1) and (2.3) specify that a starting time must be determined for each lecture. Constraints (2.2) and (2.4) ensure that students are assigned to sections of courses in which they are registered.

In this formulation, the length of lectures is not uniform, and therefore it is possible for two lectures with different starting times to be in conflict. Furthermore, the current model allows us to schedule lectures of a given course at different times on different days. However, we are now developing a variant taking into account specific patterns, e.g. Mon., Wed., Fri. for courses requiring three lectures. Finally, the set of available classrooms is partitioned into several subsets each of which includes classrooms of the same type. Of course, singleton subsets are used to denote specific classrooms, e.g. specific laboratories which are unique.

The objective function is composed of a first component $f_1(x)$ specified uniquely in terms of timetabling variables, and a second component $f_2(x,y)$ specified in terms of both timetabling and grouping variables. For the sake of reducing notational complexity, we define these functions rather than giving their mathematical expressions.

The first component $f_1(x)$ is broken down into three parts:

$$f_1(x) = f_{11}(x) + \rho f_{12}(x) + \nu f_{13}(x) \tag{1}$$

where

i) $f_{11}(x)$ is a linear assignment cost specified in terms of the availabilities and preferences of the lecturers;

ii) $f_{12}(x)$, a linear cost for exceeding classroom availabilities, is equal to the sum of classroom shortages over all periods and over all types of classrooms;

iii) $f_{13}(x)$, the cost for lecturer conflicts, is equal to the total number of times there is a lecturer-conflict during a period. $f_{13}(x)$ is a quadratic function.

The second component $f_2(x, y)$, the cost for student conflicts, is equal to the number of times there is a student-conflict during a period. This function is also quadratic.

The user controls the parameters μ, ν, ρ to guide the automatic procedures. For instance, if μ and ν are much larger than ρ and the assignments costs, then the procedures generate solutions where the conflicts are eliminated in priority. Furthermore, the context of the problems may dictate the values of the parameters. For instance, if the number of classrooms is critical, then a higher value should be given to ρ. Hence, the user may adapt the system to his specific problem by appropriately controlling the parameters.

Scheduling operator

This operator is used to schedule or reschedule a specific lecture either in automatic or interactive mode. To specify the procedure, we introduce the following notation:

i) $levelmax$ is a variable containing the maximum number of recursive calls that can be made in order to schedule a specific lecture. In interactive mode, the system will indicate all alternate solutions requiring $levelmax$ or less rescheduling of other lectures.

ii) $level$ is a variable indicating the current depth in the recursivity stack.

iii) $success$ is a variable that indicates if the procedure has been successful in scheduling the lecture currently dealt with.

In order to schedule a lecture i_1, we have to set $level$ to 0, $success$ to false, and call the procedure:

$$singletry(i_1, level, levelmax, success)$$

The procedure attempts to schedule lecture i_1 on periods j for which constraints are satisfied. If a lecture i_2 is in conflict with i_1 at a particular period, the procedure then calls itself (recursively) if $level < levelmax$:

$$singletry(i_2, level + 1, levelmax, false)$$

If the lecture currently dealt with can be scheduled directly, then $success := true$, and we have identified a possible way to schedule i_1.

In interactive mode, this procedure provides the user with a powerful tool to identify alternatives when a lecture has to be rescheduled. The number of alternatives increases with $levelmax$ and the relaxation of the constraints. If no adequate rescheduling is found, the user can increase $levelmax$ or relax the constraints further according to his own judgement. For instance, student conflicts, room shortages or lecturer availabilities may only be considered when their numbers exceed specified values.

There also exists another similar procedure that can call itself recursively two times if two lectures are in conflict at a particular period. The automatic method mentioned earlier uses these two procedures to schedule all lectures. In this mode the first alternative is selected, while in interactive mode all possible alternatives are indicated to the user.