J. Opl Res. Soc. Vol. 42, No. 3, pp. 205–216, 1991
Printed in Great Britain.

0160-5682/91 $3.50 + 0.00

A Final-exam-scheduling Package

VAHID LOTFI[1] and ROBERT CERVENY[2]
[1]School of Management, University of Michigan–Flint, USA and [2]School of Management, State University of New York at Buffalo, USA

This paper presents the implementation of a multi-phase final-exam-scheduling package at a large university. The problem as it had evolved over time is discussed and a solution is presented. The implementation was undertaken over a three-year period and involved a multi-phase approach. The multi-phase final-exam-scheduling process consists of first grouping the final exams into sets called 'blocks' in such a way as to minimize the number of students with simultaneous exams. Second, the blocks are assigned to exam days while minimizing the number of students with two or more exams per day. Third, the exam days and exam blocks within days are arranged so as to minimize the number of students with consecutive exams. Fourth, exams are assigned to classrooms so as to maximize the space utilization. New formulations and solution methods for the stated four phases are presented. The approach is well suited for a practical application and has been implemented at a large university.

Key words: exam scheduling, heuristics, optimization

INTRODUCTION

Final-exam-scheduling is an ongoing, real and troublesome process that all institutions of higher education are faced with several times a year. To all parties involved, it is most desirable to be able to develop the schedule as early in the semester as possible while retaining as much flexibility as possible. Students like to be able to make travel arrangements for the end of the semester in time to take advantage of discounts on fares, etc. (In this paper the term semester will be used for the term of instruction. While institutions vary in the actual system used, such as quarter, summer, trimester, etc., all can be conceptually related to the concept of semester.) Faculty also like to make their arrangements for both the exam period and whatever intersession gap exists between periods of instruction as early as possible. Finally, administrators like to have the process out of the way as expeditiously as possible.

The ideal time to have the final exam schedule available would be at registration time, but this requires knowing which courses are going to require a final exam at the time of registration. While this may be possible at some institutions, it is not possible at State University of New York at Buffalo (SUNYAB) where the current implementation was undertaken. At SUNYAB many factors enter into the decision to offer a final exam in any given course. First, faculty are not required to give final exams, and some courses are offered in a format that does not necessitate one. Some courses have a term paper instead, some have a project and some are seminars that do not need a final exam. Control over this is left to the departments and instructors. Also, SUNYAB does not have the luxury of having sufficient classroom space to allow all courses to be assigned a room which may or may not actually be used during the final exam period. SUNYAB policy calls for an exam to be offered in a room that has twice the seating capacity of the exam or splitting the exam across multiple rooms to provide the necessary space. This provides sufficient space for the student to work while allowing the instructor room to monitor the exam. Finally, not all classes originally scheduled at pre-registration time actually meet in any given semester and not all instructors are known until the semester begins. This is especially true for the fall semester as the pre-registration takes place in the late spring while hiring decisions for the fall are still in process.

This paper presents the development and implementation of an exam scheduling package at a large university (SUNYAB). The package consists of a set of heuristic procedures designed to solve the various phases of the problem. First, the situation as it existed prior to the intervention is discussed; next, the goals the system is trying to achieve; then the solution approach and the rationale for the directions chosen to solve the problem. Finally, the programming approach and report generation are presented, followed by a summary of the process.

Correspondence: Vahid Lofti, School of Management, The University of Michigan-Flint, Flint, Michigan 48502-2186, USA.

HISTORIC SITUATION

The previous scheduling process was an unnecessarily complex set of activities undertaken by the exam-scheduling office and the university computer services. The process consisted of eight steps, some manual and some computerized. It had arisen over a 20-year period of system evolution and reflected the piecemeal patches over time in its final complexity.

The computer programs were in three different programming languages, were last overhauled about 10 years ago, and had quirks in them that just seemed to be there. For example, during the spring of 1988, three of the final exams that were entered into the package were never scheduled and simply disappeared. No-one was quite sure why. In addition, whenever the package developed an unrecoverable error, the university computer services had to bring a consultant from another city to alleviate the problem. This arrangement was too costly and time-consuming.

The first step in the eight-step process involved preparing the list of courses having final exams. This step began immediately after the last day on which courses could be dropped, and consisted of three activities that were performed manually. The second step involved data entry and edit. The first part consisted of creating a data file and entering the registration numbers. This was undertaken by the scheduling office. It was then followed by running an edit-check program at the computer centre with a report sent back to the scheduling office. Step two took about $3\frac{1}{2}$ weeks to complete.

Step three involved combining the course registration file with the student registration file to create a data file containing student social security numbers for each course to be examined. Step four generated the student conflict matrix. A report was then sent to the scheduling office. The scheduling office would then notify the computer services with regard to the number of exam days and the number of periods per day.

Step five was the actual assignment of exams to the available periods. The package utilized an algorithm due to Broder[1] to perform the assignment. The algorithm was run three times with different seed numbers. Each run consisted of 100 schedules. The program retained the best (the one with a minimum number of students with simultaneous exams) of the 100 schedules for each run. A conflict report was then generated which allowed the scheduling office to check for (1) the number of students with two exams at the same time; and (2) the number of students with back-to-back exams. They then decided which, if any, of the three schedules to use. Often this step was repeated several times until a 'good enough' schedule was obtained.

Step six involved developing a room file containing information with regard to each room's seating capacity, availability of audio-visual equipment, etc. It also included the assignment of the final exams to the available rooms. In this case a greedy algorithm was utilized to solve the room-assignment problem. After the completion of the room-assignment problem, a report was generated which showed the actual final-exam schedule with times, rooms, and the number of students participating in the exam. Step six also reported the number of students with simultaneous exams.

Step six was complemented by an additional component, allowing various departments to request the rescheduling of certain exams. This included inclusion of exams left out of the original schedule for some reason. The scheduling office used the information developed in step six to determine the effect of moving an exam to a given period or adding an exam to a given period.

Step seven consisted of checking the room assignments. Actual availability, grouping of rooms, split exams and handicap access were all monitored in this step to identify potential problem areas. Step eight involved printing and distributing the schedule.

GOALS

The previous exam-scheduling system resided on a mainframe computer which had been on a lease from a large computer company. Upon the termination of the lease, scheduled for 48 months after the start of this project, the university's administrative computing had to be moved to a newly purchased IBM 3084. As a part of the university's administrative computing, the final exam scheduling package had to be moved as well. However, owing to the obsolescence of the previous system, the university's administration decided against moving the previous final-exam-scheduling system to the new computer and began a search for other alternatives.

One author (Lotfi) had worked on the final-exam-scheduling problem for several years and the other had extensive experience in systems analysis and design. This combination enabled the authors to form a team suitable for developing the new system. Consequently, a proposal for developing a new exam-scheduling package for the university was well received by the administration. The administration awarded the authors with two years of graduate assistance support and guaranteed support from the staff of the computer services. In consultation with the university computer services, the administration stated several goals that had to be satisfied, which are described below.

After careful consideration and discussion with major administrative parties involved (the Vice President for University Services, who has overall responsibility for facilities usage and therefore exam scheduling; the computer centre, which must maintain and run on a production basis; and the administrators, who must interface between the schedule and the faculty/department structure), it was agreed that the scheduling package had to incorporate the following set of characteristics.

A. Solution quality

Clearly, the package had to produce good-quality schedules. Quality was measured in terms of the number of students having various types of conflicts. In this case, a three-dimensional scale consisting of (1) the number of students with simultaneous exams (first-level conflict), (2) the number of students with two or more exams per day (second-level conflict) and (3) the number of students with consecutive exams (third-level conflict) was developed. The three dimensions were prioritized according to the order presented above. That is, first-level conflict was given the highest priority and the third-level conflict was assigned the least priority.

B. Efficiency

Efficiency was measured in terms of core and disk storage space requirements and speed. The package was to require the least amount of disk space possible (for input files as well as possible scratch files) and be reasonably fast. Core storage was not a major factor because the university mainframe computer utilized virtual memory technology.

C. Flexibility

The administration had expressed an interest in having the capability of performing sensitivity analysis in the form of varying the number of exam days and/or the number of periods per day. The package therefore had to be flexible enough to provide for such changes.

D. Ease of maintenance

Once the package was complete, the university computer services had the responsibility for maintaining it on an ongoing basis. Therefore, the package had to be easy for them to maintain and run.

E. User friendliness

The package had to be easy to work with. The input data had to be easy to prepare and the output easy to understand.

PROPOSED SOLUTION APPROACH

Although the final-exam-scheduling problem has been studied extensively, until recently little attention was given to the development of schedules that would include some degree of student comfort (beyond the minimization of first-level conflict). Carter[2] presents a survey of practical applications of examination-timetabling algorithms. Several successful applications are reported. However, in almost all of the stated approaches, the primary objective has been to minimize the first-level conflict. Certain methods had incorporated additional side constraints such as space limitation, pre-assignment of certain exams, etc.

Arani and Lotfi[3] suggest a three-phase approach for developing final-exam schedules that incorporates several measures of student comfort. Here, the concept of a multi-phase solution approach is also utilized, but differs in two important aspects. First, a fourth phase is added and the assignment of the final exams to classrooms is included. Second, the formulations of the first and second phases are quite different and are more suited for a practical application. In the remainder of this section the various phases of the scheduling process are described, along with the proposed solution methods as they were implemented at SUNYAB.

The proposed scheduling process consists of four phases. In the first phase, all of the final exams are assigned to a predetermined number of exam blocks. The second phase consists of grouping the exam blocks into days (three blocks per day). The third phase involves the arrangement of exam days as well as the exam blocks within exam days. Finally, the fourth phase consists of assigning the final exams to classrooms.

Phase I: assigning the final exams to exam blocks

This problem is best described by utilizing graph theory. Let $G = (V, E)$ be a graph in which each vertex $v \in V$ represents a final exam and let $N = |V|$ be the number of final exams. An edge $(i, j) \in E$ represents the existence of at least one student taking both exams i and j. Associated with each edge $(i, j) \in E$, a_{ij} is defined to be the number of students participating in exams i and j and let $a_{ij} = 0$ for $(i, j) \notin E$. Two vertices i and j are said to be adjacent if the edge $(i, j) \in E$ (i.e. $a_{ij} > 0$).

Most of the traditional final-exam-scheduling algorithms can be classified into two categories. First, those that formulate the problem as a graph colouring problem (GCP). A colouring of G is an assignment of colours (here representing exam blocks) to the vertices (here representing final exams) of G such that no two adjacent vertices share the same colour. GCP involves the development of a colouring of G (with minimum number of colours). This approach produces a conflict-free schedule (i.e. a schedule with no student having to participate in simultaneous exams) but the number of colours and therefore the number of available exam-blocks is not restricted. GCP has been studied extensively and shown to be NP-hard (Aho *et al.*[4]). Several methods, heuristics as well as exact, have been developed within the last two decades to solve this problem (see Brelaz,[5] Carter,[6] Dunston,[7] Lotfi and Sarin,[8] and Leighton[9]).

The second approach involves formulation of the problem as a quadratic assignment problem (QAP) without column constraints. In this case, the number of available exam-blocks is considered to be fixed and pre-specified. The objective is to assign all of the final exams to blocks so as to minimize the number of students with simultaneous exams. QAP has also been studied extensively and has been shown to be NP-hard (Burhard and Stratman,[10] and Murtagh and Jefferson[11]).

The QAP formulation of the phase one problem and its extensions have been used by many researchers for practical applications of final-exam scheduling (see Broder,[1] Cole,[12] Desroches *et al.*,[13] Grimes,[14] Hall and Acton,[15] and Tripathy[16]). Although all of these formulations employ the same objective function (i.e. to minimize first-level conflict), they differ in the type of constraints they incorporate. A common occurrence is to pre-assign a certain set of exams (see Leighton,[9] Mehta,[17] Peck and Williams[18] and White and Chan[19]). Another common restriction is to limit the size of the classroom (see Cole,[12] Desroches *et al.*,[13] White and Chan[19] and Wood[20]). Other constraints include where certain exams are consecutive, where certain exams are held only in the morning, where a minimum number of three exams occur in a row, and where certain exams do not precede others.

Here the phase one problem is also formulated as a QAP. However, after consulting with the scheduling co-ordinator, two additional constraint sets had to be included. The first set of constraints placed a limit on the number of exams that could be assigned to an exam block. This limit is predefined and is the same for all blocks (in our model tentatively set at 100). Clearly, this constraint set is a consequence of a limited number of available classrooms in each period. Among other things, one important aspect of this constraint set is its impact on the phase four problem, which involves the assignment of the final exams to available classrooms in each period. The second set of constraints prevents two or more exams administered by the same instructor from being scheduled simultaneously. Mathematically, the problem may be formulated as shown below.

Let $x_{ij} = 1$ if exam i is assigned to exam block j and 0 otherwise. Then, the phase one formulation is given as:

(P1)
$$\min z = 2z' = \sum_{j=1}^{P} \sum_{i=1}^{N} \sum_{k=1}^{N} a_{ik} x_{ij} x_{kj} \tag{1}$$

subject to

$$\sum_{j=1}^{P} x_{ij} = 1 \qquad \text{for } i = 1, 2, \ldots, N \tag{2}$$

$$\sum_{i=1}^{N} x_{ij} \leqslant Q \qquad \text{for } j = 1, 2, \ldots, P \tag{3}$$

$$x_{ij} + x_{kj} \leqslant 2 - b_{ik} \qquad \text{for } i = 1, 2, \ldots, N \tag{4}$$
$$k = 1, 2, \ldots, N$$
$$j = 1, 2, \ldots, P$$

$$x_{ij} = 0, 1 \qquad \text{for all } i \text{ and } j, \tag{5}$$

where N = the number of final exams;
P = the number of available blocks;
a_{ik} = the number of students taking exam i and exam k;
b_{ik} = 1 if exam i and exam k have the same instructor and 0 otherwise;
Q = the upper limit on the number of exams that can be assigned to the same exam block (maximum number of classrooms available in each period).

In the objective function (1), z' is interpreted as the total number of students having simultaneous exams and is to be minimized. Constraint (2) ensures that each exam is assigned to only one block. Constraint (3) forces the number of exams assigned to the same block to be less than or equal to Q. Constraint (4) prevents two exams with the same instructor being scheduled simultaneously. And finally, constraint (5) restricts the decision variables to be integer, 0 or 1.

Burhard and Stratman[10] provide a good review of the numerical investigation on QAP. Carlson and Nemhauser[21] investigated the special structure of the problem in absence of constraints (3) and (4). They propose an efficient heuristic solution method to solve the problem without the stated constraints. Arani and Lotfi[3] improved Carlson and Nemhauser's heuristic by implementing a better starting solution. They provide computational results showing the superiority of their algorithm over two other heuristics. Here a revised version of Arani and Lotfi's method is utilized to solve the phase one problem. The revisions facilitate enforcing the two additional constraint sets. The revised algorithm is presented in Appendix A.

Phase two: assigning the exam blocks to exam days

One way to assign the exam blocks to exam days is to assign the blocks sequentially, starting with the first exam block. This approach, which in fact has been used in the existing exam scheduling package, usually produces undesirable schedules in terms of the second- and third-level conflicts. When the objective is to minimize the number of consecutive exams, the problem may be formulated as a travelling salesman problem (TSP). This formulation assumes that the last period of one day is adjacent to the first period of the next day. In the TSP formulation, the cities represent the exam blocks and an optimal salesman tour constitutes the arrangement of the P exam blocks with a minimum number of students having consecutive exams. Colijn[22] uses the TSP formulation and compares the results with the sequential assignment. A reduction of 30% in the number of students with consecutive exams is reported. White and Chan[19] present a computer program, implementing an algorithm by Peck and Williams[18] to assign the exams to a fixed number of blocks. They proceed by using a travelling salesman problem to assign the exam blocks to exam days.

The TSP formulation of phase two has a drawback in that the optimal solution will include many students with exams in every other period. This characteristic minimizes the number of students with consecutive exams but results in many students with two or more exams per day.

Further, the last period of one day may not necessarily be considered adjacent to the first period of the next day. Arani *et al.*[23] propose a set-covering-type integer programming formulation for assigning the exam blocks to days while minimizing the number of students with two or more exams per day. They suggest an efficient Lagrangian relaxation-based branch-and-bound method for solving the integer program. Although Arani's solution method is exact and produces an optimal solution, in consulting with the personnel responsible for developing and maintaining the code, the method was considered too complex for implementation and maintenance. Consequently, less complex and more easily implemented problem formulations and/or solution methods had to be explored. Below, a new phase two formulation and a heuristic for solving it are presented.

The problem of assigning the exam blocks to exam days can be formulated as a (QAP) with column constraints (Lotfi[24]). That is, let $y_{ij} = 1$ if block i is assigned to day j and 0 otherwise. Then, the QAP with column constraints is given as:

(P2)
$$\min s = 2s' = \sum_{j=1}^{D} \sum_{i=1}^{P} \sum_{k=1}^{P} c_{ik} y_{ij} y_{kj} \tag{6}$$

subject to

$$\sum_{j=1}^{D} y_{ij} = 1 \qquad \text{for } i = 1, 2, \ldots, P \tag{7}$$

$$\sum_{i=1}^{P} y_{ij} = PD_j \qquad \text{for } j = 1, 2, \ldots, D \tag{8}$$

$$y_{ij} = 0, 1 \qquad \text{for all } i \text{ and } j. \tag{9}$$

where D = the number of exam days;

c_{ik} = the total number of students with exams in both blocks i and k ($c_{ii} = 0$ for all i);

PD_j = the number of blocks available in day j.

Hence, the objective function (6) represents the number of students with two or more exams per day and is to be minimized. Constraint (7) ensures that every block is assigned and constraint (8) forces the number of blocks assigned to day j to equal PD_j (when P is a proper multiple of D, $PD_j = P/D$ for all j). (QAP) with column constraints is also known to be a difficult problem to solve (see Carlson and Nemhauser[21]). Lotfi[24] suggests a heuristic solution procedure for solving the (QAP) with column constraints. The heuristic procedure has been shown to produce 'good'-quality solutions within a reasonable execution time. As mentioned earlier, the rationale for using the heuristic method goes beyond efficiency of the solution or optimization of the results. After consultation with the personnel who will be responsible for maintaining the system, it was decided that the code for the heuristic method would be 'easier' for a programmer to understand than the exact solution method. Among other things, the exact method requires implementations of the branch-and-bound procedure, Lagrangian relaxation and the simplex method, all in COBOL programming language. The trade-off in terms of increase in the number of conflicts likely to occur (because of the non-optimality of the method) was deemed to be less important than the cost of having to maintain the exact solution code. The details of the heuristic, as stated in Lotfi,[24] is presented in Appendix B.

Phase three: arranging the exam days and exam blocks within days

Phase III consists of an optimal arrangement of exam days and exam blocks within days in such a way as to minimize the number of students with consecutive exams. The problem of finding an optimal arrangement of exam blocks within a given day can be formulated as a TSP. However, when the number of blocks per day is three ($P/D = 3$), as is the case with the SUNYAB problem, the problem is trivial. That is, the optimal arrangement is obtained by assigning the most conflicting exam blocks to the first and the third periods. An optimal arrangement of the exam days so as to minimize the number of students with exams in the last period of one day and the first period of the next day can also be obtained via a TSP. In this case, each exam day corresponds to a city and the weight w_{ij} for arc (i, j) is the number of students taking exams during the last period of day i and the first period of day j. Since the first exam day is not predetermined, an additional

fictitious city $D+1$ is introduced with arc costs $w_{D+1,j} = w_{j,D+1} = 0$ for all j. An optimal salesman tour then represents the optimal arrangement of exam days. A heuristic commonly referred to as the 'next nearest city' (see Papadimitriou and Steiglitz[25]) is used to solve the TSP problem. The next-nearest-city solution approach results in an arrangement of exam days that approximately minimizes the number of students with consecutive exams.

Phase four: assigning the final exams to classrooms

The last phase of the multi-phase scheduling process consists of assigning the final exams to classrooms so as to maximize the space utilization. The university regulation at SUNYAB requires that a final exam be held in a classroom with a seating capacity of at least twice the number of participants. The previous system utilized a computer program that sequentially assigned the final exams to classrooms. That is, it assigned the first final exam (on the list) to the first available room (with sufficient capacity), the second exam to the second room, and so on. When it encountered a final exam for which none of the remaining rooms has enough capacity, the program split the exam into as many sections as were needed to fit into the remaining classrooms. At times, the program, not using any optimization at all, produced schedules with many split exams that otherwise would have fitted into the available space without needing to be divided. More importantly, the program did not recognize the location of the rooms and split the exams into rooms located on two different campuses. This process had become quite inconvenient because many departments offer large section courses that end up having split exams. Split exams are difficult for both the faculty and the students. First, additional staff are required. Second, the absence of the course instructor from the other sections is inconvenient to students because they may have certain questions to which only the instructor can respond. Third, if the instructor recognizes that there is an ambiguity in the exam, the point can be clarified quickly to the entire class only if they are all present.

When the objective of the phase four problem is to maximize space utilization, in the absence of any side constraints, the problem may be formulated as a minimum-weighted matching problem in a bipartite graph. That is, each final exam corresponds to a node in the first set and each classroom corresponds to a node in the second set. An edge (i, j) connects the final exam i to classroom j if classroom j has enough seating capacity for exam i. The weight associated with the edge (i, j) is the difference between twice the seating capacity of room j and the number of students taking exam i. An edge (i, j) in the optimum matching represents the assignment of the final exam i to classroom j.

The problem of assigning the final exams (scheduled in a given period) to available classrooms so as to minimize the number of split exams can be formulated as a non-linear integer program. That is, let $q_{ij} = 1$ if exam i is assigned to room j and 0 otherwise. Let p_{ij} be the number of students from exam i which are to be assigned to room j. The mathematical model is given as

(P4)
$$\min v = \sum_{i=1}^{N_t} \sum_{j=1}^{R} q_{ij} - N_t \tag{10}$$

subject to

$$\sum_{i=1}^{N_t} q_{ij} \leqslant 1 \qquad \text{for } j = 1, 2, \ldots, R \tag{11}$$

$$2p_{ij} \leqslant q_{ij} C_j \qquad \text{for } i = 1, 2, \ldots, N_t \quad j = 1, 2, \ldots, R \tag{12}$$

$$\sum_{j=1}^{R} p_{ij} = E_i \qquad \text{for } i = 1, 2, \ldots, N_t \tag{13}$$

$$\sum_{k=1}^{R} r_{kj} q_{ij} q_{ik} \leqslant 0 \qquad \text{for } i = 1, 2, \ldots, N_t \quad j = 1, 2, \ldots, R \tag{14}$$

$$q_{ij} = 0, 1, \; p_{ij} \in \{0, 1, \ldots, E_i\} \qquad \text{for all } i \text{ and } j, \tag{15}$$

where N_t = the number of exams scheduled in period t;
R = the number of available classrooms;
C_j = the seating capacity of room j;
E_i = the number of students in exam i;
$r_{kj} = 1$ if rooms k and j are not on the same campus and 0 otherwise.

(10) is the number of additional rooms required beyond the number of exams (because of splitting) and is to be minimized. (11) forces each room to be assigned to at most one exam. Constraint (11) is applicable to institutions that do not allow more than one exam in a room at the same time. This requirement may not apply to other universities, such as most universities in the UK. (12) ensures that the seating capacity is twice the number of students scheduled into that room. It also sets the binary variable q_{ij} to 1 if exam i is to be assigned to room j. (13) forces all of the participants in exam i to be assigned. (14) ensures that when an exam is split into two or more rooms then the assigned rooms are located on the same campus. (P4) is a non-linear integer programming problem with $N_t * R$ continuous variables, $N_t * R$ binary variables, and $2N_t * R + N_t + R$ constraints. The size of this problem is too large for a practical application. For example, the available classroom space at SUNYAB at present consists of 160 classrooms with varying seating capacities. Phase one can schedule as many as 100 exams in the same period, which would result in a problem with 16,000 continuous and 16,000 binary variables and over 30,000 constraints. Clearly, even if this problem could be solved by an exact method it would require extensive computational requirements. Consequently, a heuristic procedure was developed to solve (P4) (see Appendix C). This heuristic is relatively easy to understand and is suitable for practical application. An intuitive exposition of the method is presented below.

The procedure is designed to assign the final exams, scheduled in a given period, to the available classrooms so as approximately to minimize the number of split exams. Further, the method will not assign split exams to classrooms located on different campuses. It begins by sorting the classrooms in decreasing order of capacity and the exams in decreasing order of number of participants. It then tries to assign the next exam (beginning with the first one) to the first available classroom. If the number of participants in the exam is less than or equal to half the capacity of that classroom then it assigns the exam to the room and removes the exam and the classroom from the lists. Otherwise, it assigns sufficient students from the exam to fill the room and searches for another room to assign the remainder. This is done by scanning the list of available rooms, from bottom to top, searching for a room with twice the seating capacity of the remaining students, which is located on the same campus. The reason for searching the list from bottom to top is to use a secondary room which is as small as possible (to avoid wasting the capacity). Where the remaining students do not fit into the second remaining largest room, this room is used to fit as many students as possible and the list is searched for a third room.

It should be noted that in order to be able to assign all of the final exams in a given period to classrooms, the total seating capacity of all of the classrooms must be twice the total number of students in that period. This is the reason for placing an upper limit on the number of final exams assigned to the same block. At SUNYAB this upper limit was set at 100, which ensured that sufficient seating capacity would be available for every period. Where the available seating capacity does not satisfy the demand, some of the exams can be moved to other periods. However, the rescheduling of the exams should be done in such a way that the possible increase in the first-level conflict would be minimal. As will be discussed later, a decision support tool was provided which would enable the scheduling coordinator to do just that.

IMPLEMENTATION AND REPORT FORMATS

The stated multi-phase scheduling system was programmed on an IBM-3084 mainframe computer in COBOL programming language within the CMS environment. COBOL was utilized because of the strong desire on the part of the group responsible for administrative programming to have a common language in all administrative computing. COBOL had been selected earlier by them as the language of choice. A collection of nine programs were developed to perform the necessary data manipulation, file processing, and optimization required by the proposed system.

The system requires three source data files and generates several reports. The three data files consist of

(1) the course registration file containing course identification numbers and the social security numbers of students enrolled in each course;
(2) the course file containing course identification numbers and the social security numbers of the associated instructors;
(3) the room file containing available rooms with associated capacities and their locations.

The system was designed such that making changes to a generated schedule could be achieved with relative ease. In particular, two features were incorporated into the system for this purpose. First, the system was designed so that if the scheduling co-ordinator decided to reassign some exams from their present blocks (periods) to other ones then the first phase need not be executed again. Instead, an editor can be used to make changes to the file containing the current assignments and then the remaining three phases run once more. Second, the system generates several reports that provide for making small changes to a generated schedule without the need for executing any of the programs again. Below, a summary of the various reports generated by the system is presented.

Phase one report

The first report, generated at the end of phase one, consists of three components. The first component indicates the number of students with simultaneous exams. The second consists of an alphabetical list of courses (final exams) and their present block assignments. The third component is a sensitivity analysis table indicating, for each exam, the present block assignment and potential increase in the total conflict by reassigning that exam to another block. In addition, this table also indicates whether the instructor for a given exam is offering another exam assigned to a different block. This is achieved by placing an asterisk (*) next to the potential conflict increase of that block.

Phase two report

The report produced at the end of this phase presents the number of students with two or more exams per day and the matrix of block assignments.

Phase three report

The report produced at the end of phase three presents the actual schedule indicating the days, the periods within the days, and the courses assigned to them.

Phase four report

The last report produced at the end of phase four consists of two parts. The first part indicates the room assignment for each exam, grouped by day and by periods within days. The second part consists of a list of available rooms by periods (for each day), indicating the current course assignment or a blank entry if no exam has been assigned to the room for that period.

RESULTS AND CONCLUSIONS

At present, the new system is being run in parallel with the previous system while it is gradually being interfaced with other administrative information systems components. Comparisons between the schedules for the Fall of 1988 semester indicated that the performance of the new system is superior to that of the old one. In particular, both packages were used to schedule 858 final exams, involving 11,331 students, into 5 days each consisting of 3 periods for a total of 15 periods. The previous system resulted in 246 students having simultaneous exams, whereas the new package had 182. The schedule generated by the previous system had 1434 students with consecutive exams, the new method had 1376. The previous package does not report the number of students with two or more exams per day. The new package resulted in an average (per day) of 655 students with two or three exams. The previous package assigned the 858 exams into rooms such that 315 exams were split into 2 rooms, 43 exams were split into 3 rooms, and 5 exams were

split into 4 rooms. The new package resulted in 61 exams split into 2 rooms and no exams split into 3 or 4 rooms.

In addition to its superior performance with respect to the quality of the schedule produced, the new system is also quite efficient. The package requires approximately one hour of execution time from start to finish (all four phases). The previous system required extensive computing time and core memory. It had to be run overnight with no other jobs running simultaneously. Its execution time (start to finish) required several hours, depending on the number of exams and the number of trial runs.

The discussion above applies a combination of algorithmic and heuristic procedures to a problem faced by a large number of institutions. It creates an exam schedule that attends to the comfort level of all parties involved in an efficient and effective manner while incorporating elements of decision support systems (DSS) into the output. The issues which constitute the comfort level include the following:

1. Minimize the number of students with simultaneous exams.
2. Minimize the number of students with two or more exams per day.
3. Minimize the number of students with consecutive exams, both within days and between days.
4. Minimize the number of split exams (exams assigned to more than one room).
5. Produce DSS reports to allow immediate changes to be made by scheduling personnel.
6. Implement in a language which can be readily supported by administrative computer centre personnel.
7. Utilize procedures that can be solved in a reasonable time without requiring excessive computer resources.
8. Rationalize procedures involved with the process so that administrative personnel can create the final-exams schedule early in the semester easily.
9. Provide information to the consumer in time to make decisions about how to react to the process.

The results of this effort will help to improve the quality of university life at this institution by reducing the irritating effects of the process on the people involved while providing earlier information to the consumers of the product so they may make more informed decisions about how to plan their end-of-semester activities. While not providing exact solutions to the problem, the solution balances the trade-off between timeliness of the solution and ease of implementation.

APPENDIX A

Phase One Heuristic

To begin, all of the final exams (nodes of G) are sorted in a descending order of weighted degrees. The weighted degree of a node is defined as: $w_i = d_i \sum_k a_{ik}$, where d_i is the degree of node i and $\sum a_{ik}$ is the total number of students taking exam i and all other exams in conflict with exam i. The reason for using the weighted degree is that it incorporates both the number of students and the number of exams in conflict with it to give a sense of difficulty of scheduling it. Therefore, a colouring algorithm schedules the more difficult (more conflicting) exams first. Let $v[1], v[2], \ldots, v[N]$ be the N final exams ordered as stated above. The steps of the heuristic are:

Step 0. Let B_i be the set of final exams assigned to block i; initialize $B_i = \phi$ for all $i = 1, 2, \ldots, P$. Choose the first exam from the top of the ordered list, say $v[1]$, and assign it to block 1 (i.e. $x_{v[1],1} = 1$), let $B_1 = \{v[1]\}$. Set $n = 1, j = 1$, and $V = \{1\}$. Go to Step 1.

Step 1. If $n = N$, all exams have been scheduled, go to 4. Otherwise, let W be the set of potential unfilled blocks given as:

$$W = \{t \mid a_{v[n+1],k} = 0 \quad \text{and} \quad b_{v[n+1],k} = 0 \qquad \text{for all } k \in B_t, |B_t| < Q, t \in V\}.$$

If $W = \phi$, no potential block exists, go to Step 2. Otherwise, let $k = \min\{t \mid t \in W\}$ be the first unfilled block with no conflicting exams or instructors with $v[n+1]$. Assign $v[n+1]$ to block k, i.e. let $B_k \leftarrow B_k \cup \{v[n+1]\}$, $x_{v[n+1],k} = 1$, $n \leftarrow n + 1$, and repeat Step 1.

Step 2. If $|V| < P$ go to Step 3. Otherwise, let $U = \{t \mid b_{v[n+1],k} = 0 \text{ for all } k \in B_t \text{ and } |B_t| < Q\}$. U is the set of unfilled blocks with no administrator conflicts with $v[n+1]$. Compute

$$c_{t'} = \min\left\{\sum_{k \in B_t} a_{v[n+1],k},\ t \in U\right\},$$

assign exam $v[n+1]$ to block t', i.e. let $B_{t'} \leftarrow B_{t'} \cup \{v[n+1]\}$, and $x_{v[n+1],t'} = 1$. Let $n \leftarrow n+1$ and go to Step 1.

Step 3. Let $j \leftarrow j+1$ and $V \leftarrow V \cup \{j\}$. Assign exam $v[n+1]$ to block j, i.e. let $B_j \leftarrow B_j \cup \{v[n+1]\}$, and $x_{v[n+1],j} = 1$. Let $n \leftarrow n+1$ and go to Step 1.

Step 4. All exams are scheduled. Use this solution, $X = [x_{ij}]$, as a starting solution and apply the revised version of Carlson's method for possible improvements.

APPENDIX B

Phase Two Heuristic

The algorithm begins by constructing an initial solution obtained by assigning the exam blocks to exam days sequentially. That is, the first P/D blocks are assigned to the first day, the second P/D blocks to the second day and so on. Then, the current solution is examined for possible improvements through a pairwise exchange of exam blocks. If no further improvement is possible, then the algorithm stops and the current solution is used as input to the third phase. Steps of the algorithm are as follows:

Initial. Use a sequential assignment to construct an initial solution. Let $q = P/D$, and

$$y_{ik} = \begin{cases} 1 & \text{for } i = (k-1)*q+1, (k-1)*q+2, \ldots, (k-1)*q+q, \quad \text{and} \quad k = 1, 2, \ldots, D \\ 0 & \text{otherwise.} \end{cases}$$

Step 0. Let $C = [c_{ij}]$ and $Y = [y_{ij}]$, determine $R = CY$. Let $i = 0$, Flag = False, and go to Step 1.

Step 1. Let $i \leftarrow i+1$. If $i > P$ go to Step 4. Otherwise, find $r'_{iq} = \min r_{ik}$. If $y_{iq} = 0$ go to Step 2. Otherwise, repeat Step 1.

Step 2. In row i of Y find column t with $y_{it} = 1$. For each row j in which $y_{jq} = 1$ let $w_j = (r_{it} - r_{iq}) + (r_{jq} - r_{jt})$. Let $w_k = \min w_j$. If $w_k \geqslant 0$, go to Step 1, else go to Step 3.

Step 3. Let $y_{kq} = 0$, $y_{kt} = 1$, $y_{iq} = 1$, and $y_{it} = 0$. Let $R = CY$, Flag = True, and go to Step 1.

Step 4. If Flag = False, stop. No further improvement is possible. Otherwise, let Flag = False, $i = 0$, and go to Step 1.

APPENDIX C

Phase Four Heuristic

Let $v_1, v_2, \ldots, v_n$ be the n final exams assigned to period t. Let $r_1, r_2, \ldots, r_p$, be the available classrooms. Further, let E_{vi} be the number of students participating in exam v_i and C_{rj} be the seating capacity of room r_j. The steps of the heuristic are presented below.

Step 0. Sort all of the exams (assigned to period t) in a descending order of number of students, and sort all of the classrooms in a descending order of the seating capacity. Let $v[1], v[2], \ldots, v[n]$ be the ordered exams and $r[1], r[2], \ldots, r[p]$, be the ordered classrooms. Set $a_k \leftarrow 0$, $k = 1, 2, \ldots, p$, (value of 0 for a_k indicates that room k has not been assigned yet), $i \leftarrow 1$, $j \leftarrow 1$, and go to Step 1.

Step 1. If $a_j = 0$ go to Step 2. Let $j \leftarrow j + 1$. If $j > p$ stop, problem is infeasible. Otherwise, repeat Step 1.

Step 2. If $E_{v[i]} > \lceil \frac{1}{2} C_{r[j]} \rceil$, go to Step 3. Assign exam $v[i]$ to room $r[j]$. Let $a_j \leftarrow 1$, $i \leftarrow i + 1$, and $j \leftarrow j + 1$. If $i > n$, stop, all exams have been assigned. Otherwise, go to Step 1.

Step 3. Let $\delta \leftarrow E_{v[i]} - \lceil \frac{1}{2} C_{r[j]} \rceil$. Assign δ students of exam $v[i]$ to room $r[j]$, let $k \leftarrow p$, $q \leftarrow 0$, and go to Step 4.

Step 4. If $k \leqslant j$, go to Step 5. If $a_k = 1$, or $r[k]$ is not on the same campus as $r[j]$, let $k \leftarrow k - 1$ and repeat Step 4. Otherwise, go to Step 6.

Step 5. If $q = 0$, stop. There is no feasible solution. Otherwise, assign δ students of exam $v[i]$ to room $r[q]$. Let $a_q \leftarrow 1$, $\delta \leftarrow \delta - \lceil \frac{1}{2} C_{r[q]} \rceil$, $k \leftarrow p$, $q \leftarrow 0$, and go to Step 4.

Step 6. If $\delta > \lceil \frac{1}{2} C_{r[k]} \rceil$, go to Step 7. Otherwise, assign the remaining students of exam $v[i]$ to room $r[k]$. Let $a_k \leftarrow 1$, $i \leftarrow i + 1$, $j \leftarrow j + 1$, and go to Step 1.

Step 7. Let $q \leftarrow k$, $k \leftarrow k - 1$, and go to Step 4.

Acknowledgement—This research was supported in parts by a grant from the Office of Vice President for University Services, State University of New York at Buffalo. The authors wish to thank the referees for their helpful comments.

REFERENCES

1. S. Broder (1964) Final examination scheduling. *Comm. ACM* **7,** 494–498.
2. M. W. Carter (1986) A survey of practical applications of examination timetabling algorithms. *Opns Res.* **34,** 193–202.
3. T. Arani and V. Lotfi (1989) A three phased approach to final exam scheduling. *IEE Trans.* **21,** 86–96.
4. A. V. Aho, J. E. Hopcroft and J. D. Ullman (1974) *The Design and Analysis of Computer Algorithms.* Addison-Wesley, Reading, Mass.
5. D. Brelaz (1979) New method to color the vertices of a graph. *Comm. ACM* **22,** 251–256.
6. M. W. Carter (1983) A decomposition algorithm for practical timetabling problems. Working paper No. 83–86, Dept of Industrial Engineering, University of Toronto.
7. F. D. J. Dunston (1976) Sequential colorings of graphs. In *Proc. 5th British Comb. Conf., Aberdeen, 1975. Cong. Num. XV, Utilitas Math.* 151–158.
8. V. Lotfi and S. Sarin (1986) A graph coloring algorithm for large scale scheduling problems. *Comput. Opns Res.* **13,** 1, 27–32.
9. F. Leighton (1979) A graph coloring algorithm for large scheduling problems. *J. Res. Natl Bur. Stand.* **84,** 489–506.
10. R. E. Burhard and K. H. Stratman (1978) Numerical investigations on quadratic assignment problems. *Nav. Res. Logist. Q.* **25,** 129–148.
11. B. A. Murtagh and T. R. Jefferson (1982) A heuristic procedure for solving the quadratic assignment problem. *Eur. J. Opl Res.* **9,** 71–76.
12. A. J. Cole (1964) The preparation of examination time-tables using a small-store computer. *Comp. J.* 117–121.
13. S. Desroches, G. Laporte and J. M. Rousseau (1978) HOREX A computer program for the construction of examination schedules. *INFOR* **16,** 294–298.
14. J. Grimes (1970) Scheduling to reduce conflict in meetings. *Comm. ACM* **13,** 351–352.
15 A. Hall and F. Acton (1967) Scheduling university course examinations by computers. *Comm. ACM* **10,** 235–238.
16. A Tripathy (1980) A lagrangian relaxation approach to course timetabling. *J. Opl Res. Soc.* **31,** 599–603.
17. N. K. Mehta (1981) The application of a graph coloring method to an examination scheduling problem. *Interfaces* **11,** 57–64.
18. J. E. Peck and M. R. Williams (1969) Algorithm 286 examination scheduling. *Comm. ACM* **9,** 433–434.
19. G. White and P. Chan (1979) Towards the construction of optimal examination schedules. *INFOR* **17,** 219–229.
20. D. C. Wood (1968) A system for computing university examination timetables. *Comp. J.* **11,** 41–47.
21. R. Carlson and G. Nemhauser (1967) Scheduling to minimize interaction cost. *Opns Res.* **14,** 52–58.
22. A. Colijn (1979) Reduction of second order conflict in examination timetables. Working paper, Dept of Computer Science, University of Calgary, Calgary, Alberta, Canada.
23. T. Arani, M. H. Karwan and V. Lotfi (1988) A lagrangian relaxation approach to solve the second phase of the exam scheduling problem. *Eur. J. Opl Res.* **34,** 3, 372–383.
24. V. Lotfi (1986) Second level conflict resolution in scheduling problems. Working paper No. 678, School of Management, State University of New York at Buffalo.
25. C. H. Papadimitriou and K. Steiglitz (1982) *Combinatorial Optimization: Algorithm and Complexity.* Prentice-Hall, Englewood Cliffs, NJ.