A MATHEMATICAL PROGRAMMING TECHNIQUE FOR SCHEDULING COURSES AT THE SLOAN SCHOOL

by

Richard C. Ocken

B.A., University of Pennsylvania
(1983)

SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE
DEGREE OF

MASTER OF SCIENCE IN MANAGEMENT

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1987

Signature of Author: [illegible]
Alfred P. Sloan School of Management
May 15, 1987

Certified by: [illegible]
Charles H. Fine
Thesis Supervisor

Accepted by: [illegible]
Jeffrey A. Barks
Associate Dean, Master's and Bachelor's Programs

A MATHEMATICAL PROGRAMMING TECHNIQUE FOR SCHEDULING COURSES AT THE SLOAN SCHOOL

by

Richard C. Ocken

Submitted to the Alfred P. Sloan School of Management on May 15, 1987 in partial fulfillment of the requirements for the Degree of Master of Science in Management

ABSTRACT

Many schools and universities have been able to schedule resources more efficiently by using formal mathematical models. The course schedules generated by such models may be preferred by all of the school's major constituencies: faculty, students, and the administration.

This thesis describes the development of a mathematical programming model used to schedule courses at the Sloan School. The available literature on the general timetabling problem is reviewed to determine the applicability of the various methods to the Sloan School problem. A Decision Support model based on linear programming techniques is developed. That model is distinguished from many of the others in the literature by its completeness.

Alternative schedules are generated for the actual Fall, 1987 semester. The best model-generated alternative is shown to be apparently superior to the one generated by the existing manual process. Means of comparison include an improvement in faculty satisfaction and the avoidance of conflicts between courses likely to have a high degree of overlap in student demand. I conclude that the model represents a viable approach to course timetabling at the Sloan School and I discuss recommendations for the next steps required to implement it more formally in the future.

Thesis Supervisor: Dr. Charles H. Fine

Title: Assistant Professor of Management Science

PREFACE

I would like to acknowledge the assistance and advice provided me by members of the Sloan School Faculty and Administration. Professor Charlie Fine provided the inspiration for the project, and I have benefited from his comments on my ideas and text. Professor Rob Freund helped me with some specific mathematical issues. Finally, I benefited from Professor Tom Magnanti's suggestions and political support for the survey sent to the faculty. I would like to thank all three of them for giving me so much of their time during my work on this project.

Two members of the Sloan School Administration, Hillary deBaun and David Weber, were particularly helpful in providing an understanding of the current process and policies, as well as the data for the actual test run. I would like to thank them, as well as Associate Dean Jeffrey Barks, for their help and suggestions during the project.

Other people I would like to thank include: all of the members of the Sloan Faculty and Class of 1988 who cooperated by returning the surveys I distributed; my friends at Consultants for Management Decisions for allowing me to use their laser printing facilities; and, most importantly, all of my other friends and family who have provided personal support throughout my work on this project and during my years at Sloan.

Graduations are called Commencements because they represent a beginning. In that spirit, I would like to dedicate this thesis to the members of the next generation of students in my family, my niece, Julie Ann Ocken, and my nephew, Adam Russell Schrage.

CONTENTS

CONTENTS, (continued)

I. Introduction.

Schools at all levels face a timetabling problem in assigning their resources (faculty, classes, and rooms) to weekly schedules. Recent developments in timetabling techniques and computer technology have permitted schools to schedule their resources more efficiently by using formal models embodied in computer systems. (Sabin and Winter (1983) and de Werra (1985)). Potential benefits could include the generation of schedules that are:

- Preferred by the faculty in terms of time assignments, smoothness of load, and so on;

- Preferred by the students in terms of less classroom overcrowding and fewer time conflicts between courses that may have a high degree of overlap in student demand;

- Preferred by the administration in terms of a more efficient use of physical resources, achieved by matching courses to rooms close in size.

In this thesis, I describe the application of a specific mathematical programming technique to the timetabling problem at the Sloan School of Management at MIT. A Decision Support model based on linear programming techniques is developed. This model is shown to be useful in helping a human scheduler balance preferences and interests across and within the various constituencies. Alternative schedules are generated for an actual problem. These alternatives are apparently superior to the schedule developed by the existing manual process. I conclude that a computerized approach to timetabling is viable for the Sloan School and I make suggestions for its formal implementation. Finally, it should be noted that while I have focused mainly on the specific problem at the Sloan School, my methodology is applicable to timetabling problems at other schools or in other scheduling contexts.

My thesis is organized as follows. Section II provides a description of the Sloan School problem, in terms of the political context, the current process, the impetus for the development of a new system, and the agreed-upon model scope. Section III discusses approaches to timetabling problems described in the Operations Research literature and evaluates the applicability of these approaches to the Sloan School problem. Section IV presents a detailed formulation of a mathematical programming model of the Sloan School timetable and discusses some issues raised by that formulation. Section V discusses my solution technique and my prototype implementation of that methodology for the Sloan School problem. Section VI describes an actual test case of that model for the Fall, 1987 schedule. In particular, the schedules generated by the model are shown to be apparently superior to the one generated by the existing manual process. Finally, Section VII presents my conclusions and suggestions for next steps in formally implementing the model.

There are six appendices to this thesis. They are listed in the contents and are referred to in the text at the appropriate times.

II. <u>The Sloan School Timetabling Problem</u>

II.A. <u>Overview of the Sloan Context</u>

The Sloan School of Management at MIT offers courses in several programs: Masters', PhD, Bachelors', Sloan Fellows (a one year Masters' program for managers), and Management of Technology. Most of these courses share room and faculty resources and some of the courses in the different programs may have students cross-registered among them. In addition, the School shares facilities with other departments at MIT. MIT policy to date has been to allow the Sloan School first priority to schedule courses in the rooms in its own buildings. However, to the extent that other departments may require some of the room facilities in the Sloan buildings, MIT as a whole is better off if Sloan designs its schedule to utilize its room facilities most efficiently.

The masters' program is the largest and most complex to schedule since it has the most students (approximately 200 per class in each of two classes) and the most courses (students in other programs meet some of their requirements by taking masters' courses). In addition, students in the masters' program spend only two years at the school, one of which is largely dedicated to required (CORE) courses. Thus, the students may have only one or two opportunities to take certain elective courses during their time at Sloan. The students' problem is complicated by the fact that they must fulfill requirements for a concentration in a given field (e.g., Operations Management, Finance, etc.). Most of the concentrations require a student to take two or three elective courses in that field. Some of those electives may be specified; others may be selected by the student from an approved list.

Most courses at the school meet three hours per week in two 90 minute sessions. Sloan School policy has been, wherever possible, to schedule courses within one of eight standard time slots during the week (Monday and Wednesday or Tuesday and Thursday; 9:00 to 10:30 AM, 10:30 AM to 12 Noon, 1:00 PM to 2:30 PM, and 2:30 to 4 PM). However, there are certain exceptions to this policy. First, faculty members may choose to schedule courses at off hours on Fridays or in the evenings. These courses tend to be small seminars and there has never been any problem scheduling courses at such times in the past, if the faculty member so desires. Second, some of the CORE courses may meet in certain non-standard formats. Such formats may include an additional session on Fridays and/or a course duration of only a half semester. These complications will be discussed in greater detail in Section IV, below. Third, faculty members may desire certain courses to be taught in one continuous three hour session. If such a course were to be scheduled fully within the standard hours, it would, in effect, consume two of the eight standard time slots, both from the students' as well as a facilities perspective. Sloan School policy has been to try to schedule these courses to begin at 2:30 PM so that the second 90 minutes would occur during off hours. In this way, the students involved would not lose a second standard slot by taking the course and additional room capacity would not be consumed during the peak times.

II.B. The Current Process

II.B.1. Sloan Fellows Courses

The Sloan Fellows program is designed to fit tightly into its one year time frame. In general, the program is scheduled independently of the other programs at the school, although certain faculty members may teach and some students may take courses in the Sloan Fellows program as well as in others. However, for most of this project, I have taken the schedule for the Sloan Fellows program as a given around which the other courses must be scheduled. Any exceptions to this assumption will be stated explicitly.

All other courses in the school can be considered as belonging to one of two groups: Masters' CORE courses; and electives (including Masters', PhD, Bachelors' and other courses).

II.B.2. CORE Courses

In conjunction with a subset of the faculty responsible for the CORE, the Sloan School administration schedules the CORE courses on a centralized basis. Currently, the first year students are divided into 12 sections (A through L). In each of the two semesters of the first year, the students are required to take seven courses, some of which may meet for only a half semester. The courses meet in various sizes, ranging from two sections (approximately 30 students) to six sections (approximately 90 students) meeting together at one time. (Grouping of sections will be discussed in Section IV, below). While faculty members teaching the CORE courses have some degree of input regarding their time preferences, the central administration has ultimate scheduling authority since each first year section must be guaranteed a feasible schedule, i.e., one in which the students in each section can attend all of their required courses without conflict. Faculty preferences are generally known, however, and the administration does try to accomodate them where possible.

II.B.3. Elective Courses

The current scheduling process for all other courses (electives) is a somewhat decentralized, manual process. The school is divided into three Areas (e.g., Management Science), each of which is subdivided into an average of five Groups (e.g., Operations Management). The current process is decentralized in the sense that decisions regarding which courses will be taught and at what times they will be taught are made at the group level by the individual faculty members involved. The administration serves mainly to collect the data and to make sure that sufficient room resources are available. Thus far, sufficient room capacity has been available in that the administration always has been able to assign each course to a room. However, many large courses have been assigned to small rooms due to a shortage of the larger rooms at certain popular times. The results have sometimes included overcrowding, students dropping courses they would have taken but for the room size constraints, and/or forced restrictions on class size by the faculty member using lotteries, etc. In addition, the manual nature of the system makes the process of accomodating changes in the schedule difficult for the administrative staff responsible to implement such changes.

One should note that decentralization of the scheduling function and the resulting large degree of faculty automomy is a deep-seated tradition at Sloan and at MIT as a whole. Faculty members have an extraordinary amount of say regarding their elective teaching schedules, in terms of which courses they teach and when these courses will be taught. Several studies (e.g., Sabin and Winter (1986), McClure and Wells (1986), etc.) have documented potential benefits of computerized scheduling systems in many academic environments. Such benefits may include more efficient use of resources, improved response time to changes, smoother schedules, and so on. These and other potential opportunities may exist at Sloan as well. However, the school's culture implies that in order for any new scheduling system (computer or manual) to be successfully implemented, it must first be widely accepted by the faculty. In turn, this requirement means that the system will have to continue to allow the faculty members to have a great deal of input into their scheduling assignments.

II.C. Impetus for a New System at Sloan

Students at the Sloan School are not directly involved in the process of planning course timetables. In particular, course schedules are presented to the students without any formal attempt to ascertain which course pairs are likely to have a significant degree of overlap in student demand. While there is presumably some coordination by faculty at the group level to avoid potential student conflicts, there is little formal coordination across groups. Thus, many students concentrating in Management Information Systems (MIS), say, may wish to take a course in Technology Strategy (offered by the Strategy Group). However, if the Strategy and MIS groups fail to coordinate (which may be likely since they are not even in the same Area), their offerings may be scheduled in conflict. Again, since Sloan Masters' students have only one year to devote to electives, they may have only one opportunity to take these courses. Thus, if the courses are offered concurrently, students may be precluded from taking both during their time at the school.

The Sloan School administration would like to be able to improve the scheduling process so as to decrease or eliminate time conflicts between courses that have a high degree of potential overlap in student demand. However, the current process for scheduling electives does not give the faculty members the means or incentives to be flexible in their time requests. That is, the current process asks the groups only for the one time desired for each course. As a result, courses tend to be scheduled largely by inertia, that is, by scheduling them at the same time as last year.

In fact, many faculty members may be relatively flexible or even indifferent with regard to their time preferences. Meanwhile, the manual nature of the process ensures that by the time the schedule is published, it is too late to make major changes without affecting many other students and faculty.

II.D. <u>Model Scope</u>

II.D.1. <u>Political Realities and Boundary Conditions.</u>

I have begun to discuss some of the political aspects of the scheduling process above. In this section, I will summarize four key political constraints on the development of any new scheduling system, computerized or otherwise.

i. <u>Faculty Input.</u>

While MIT and Sloan faculty members are presumably comfortable with the application of quantitative methods and computer technology to management problems, they are still used to a scheduling process in which they have a great deal of input and control. Hence, they are likely to reject any new system that is perceived to be characterized by some omnipotent scheduler (computer or human) that has ultimate authority over their teaching assignments. Such a system would be very much in conflict with the culture at Sloan and at MIT in general. Thus, any new scheduling system will have to include significant facilities to accomodate faculty input and preferences. Furthermore, the soft nature of the process implies that ideally, the system should be conceived and implemented as an aid to a decision maker (Decision Support System) as opposed to a single pass "black box".

ii. <u>System Flexibility.</u>

A fact of life at Sloan is that changes are frequently made to expected schedules due to new course offerings, visiting faculty, and so on. Any new system will have to be flexible enough to allow for incremental changes and frequent revisions and runs ("scenarios").

iii. <u>Student Preregistration.</u>

Another deep-seated Sloan/MIT tradition is the policy of allowing students a great deal of flexibility in dropping and adding courses. Students may add courses to their schedules up to five weeks into the semester, and drop them up to three weeks before the end of the semester. As a result, the administration can not hold students to their stated preregistration preferences. In turn, this policy implies that the faculty may tend to be less willing to rearrange their own schedules solely to minimize expected student conflicts calculated from data based on student preregistration requests.

iv. <u>Skewed Demand for Courses.</u>

Certain courses and faculty members are extremely popular at Sloan. Some of the most popular electives may draw nearly half of the second year class. Other faculty members often try to avoid scheduling their courses opposite such other courses, no matter what a scheduling system might say.

II.D.2. <u>Actual Model Scope.</u>

In deciding on an appropriate scope for the prototype model developed in this thesis, I considered all of the factors and characteristics of the process as discussed above. The actual model scope presented below was agreed upon by the relevant members of the administration as well as by certain key faculty leaders.

The model is essentially a vehicle for assigning given course offerings to rooms and times. That is, the faculty groups will continue to decide which courses will be offered and who will be responsible for teaching those courses. The model will <u>not</u> choose the courses to be offered nor will it make endogenous assignments of faculty to course offerings.

Within this general framework, the model will consider courses and time as follows:

i. <u>Time Slots.</u>

The model will consider only those courses that will be offered during one of the eight standard time slots as described above. It will assume that sufficient capacity and flexibility are available for courses scheduled during off hours. The model <u>will</u> schedule those courses offered within the standard week but in non-standard formats. A detailed discussion describing how such courses will be handled is given in Section IV.

ii. <u>Courses.</u>

The model will schedule all of the courses in the following programs: Masters' (CORE and elective); PhD; and Bachelors'. The model will take the Sloan Fellows courses as given. The reader may note that although the model is formally responsible for scheduling the PhD courses, in general such courses can be scheduled whenever the faculty member desires. This result stems from the fact that such courses tend to be smaller and thus pose fewer problems with regard to student conflicts and room availability. (Sloan generally has plenty of seminar rooms available.) With regard to three hour seminar courses, as a first pass the model will schedule such courses at 2:30 PM. Manual changes can be made afterward.

A detailed formulation of the Sloan School model will be given in Section IV. However, before delving into the details of that model, I will first describe approaches to the timetabling problem that have been proposed and reported on in the Operations Research literature.

III. Approaches to Timetabling in the Literature

Several different approaches to the timetabling problem have been proposed and reported on in the Operations Research literature. This section is not meant to be a comprehensive discussion of the literature; it serves only to give the reader a broad understanding of the range of past work. However, since I have borrowed ideas from some of these works in developing a prototype system for the Sloan School, it is important for me to describe the relevant aspects of these approaches, albeit briefly. If the reader desires a more comprehensive survey of timetabling, he is referred to the bibliographies of de Werra (1985) and Schmidt and Strohlein (1979).

III.A. The Formal Timetabling Problem

In order to facilitate the discussion of the literature and potential applications to the situation at the Sloan School described in Section II, I will, for future reference, first specify the general form of the timetabling problem. The problem can be formulated in the following way.

Suppose there are $i = 1,..,I$ courses to be scheduled in $j = 1,..,J$ different time periods. (For simplicity assume that the time periods are defined so that each course is assigned to only one slot.) Then we can define variables x such that:

x_{ij} = { 1 if course i is assigned to time j;

{ 0 otherwise;

Let I_r = set of courses that are compatible with room type r (e.g., sufficient capacity);

I_f = set of courses taught by faculty member f;

I_s = set of courses to be taken by class (i.e., section) s;

c_{ij} = a measure of the desirability of assigning course i to time j (e.g. faculty "utility");

$N\{r\}_j$ = the number of rooms of type r available at time j;

Then the formal problem is:

$$\max \quad \sum_{i=1}^{I} \sum_{j=1}^{J} c_{ij}x_{ij} \qquad (1)$$

$$\text{such that} \quad \sum_{j=1}^{J} x_{ij} = 1 \qquad \text{all } i, \qquad (2)$$

$$\sum_{i \in I_r} x_{ij} <= N\{r\}_j \qquad \text{all } j,r \qquad (3)$$

$$\sum_{i \in I_f} x_{ij} <= 1 \qquad \text{all } j,f \qquad (4)$$

$$\sum_{i \in I_s} x_{ij} <= 1 \qquad \text{all } j,s \qquad (4)$$

It is instructive to discuss the formulation and its underlying assumptions in some detail. The objective term (1) is a measure of the total satisfaction derived from the schedule, defined as the sum of individual utility contributions c_{ij}. Now, the Arrow Impossibility Theorem demonstrates that no group (social) utility function can be designed so as to meet several basic (and reasonable) criteria: nondictatorship, Pareto criterion, etc. (See, _e.g._, Graham (1980) or any other welfare economics text.) The problems lay in the comparisons of individuals' utility functions: should every individual receive an equal weighting; how can one individual's rating be compared to another's, etc. Utility theory assumes that utility functions are ordinal only and that cardinal values can not be assigned to bundles of goods. Thus, it is impossible to compare or add different individuals' utility functions. Still, for lack of a better method, developers of most timetabling models do tend to use an additive choice utility function. That is, the total satisfaction produced by a timetable is defined to be the (unweighted) sum of the individual utilities of the relevant players.

Constraints (2) guarantee that all courses are assigned to one slot j during the week. I assume for purposes of this discussion that the j's have been defined as slots so that each class need only be assigned one slot during the week. (e.g., Monday and Wednesday, 9:00 to 10:30.) If courses were to be assigned a variable number of hours, the constraints would be the same except that the right hand sides would be the appropriate number of hours required by each course.

Constraints (3) ensure that room capacity constraints are met. I have assumed that the rooms are grouped by size (or available equipment) into R groups (Mulvey, 1982). This assumption is appropriate if the rooms can be grouped into like categories. An alternative approach would be to maximize the number of filled seats such that all classes can fit in the rooms they are assigned. As we shall see in Section IV, rooms at the Sloan School can be roughly grouped into four size categories so that the grouping method is appropriate. For a discussion of the alternative treatment, the reader may consult Glassey and Mizrach (1986) or Mulvey (1982).

It is significant to note that equations (2) and (3) define a unimodular matrix. Thus, by the integrality property of networks, (Bradley, Hax, and Magnanti (1977)), the application of the traditional simplex method to those equations will yield an integer solution. In particular, equations (2) enforce the upper bound of 1 on each variable x_{ij} so that the simplex method will yield an appropriate binary solution. However, in reality most timetabling problems have additional resource constraints such as (4) and (5) which complicate the network. Thus, the simplex method cannot be guaranteed to yield an integer solution and specific integer programming techniques must be used. (A more detailed discussion of these issues will be given in Section III.D.)

Constraints (4) and (5) ensure other typical facets of schedule feasibility. Constraints (4) guarantee that no faculty member will be scheduled to teach more than one course at any given time. Similarly, constraints (5) guarantee that no group of students (a section) will be required to take more than one course at one time. Such constraints may not exist for an undergraduate program, say, where students are not grouped into specific sections. However, most graduate programs (e.g., law, business, etc.) do use some kind of a section system with various required courses. As discussed in Section II, the Sloan CORE program does divide students into sections which take required courses together.

Finally, we should recognize that the formulation presented above represents only the minimum requirements for schedule feasibility. Other constraints could exist. For instance, we might wish to include constraints to ensure that no faculty member teach back to back, that two courses i and i' must (not) be taught on the same day, etc. Section IV will present the detailed formulation for the Sloan School timetable, which will include these and other constraints.

For purposes of the following discussion, I have grouped the past work on timetabling into three broad categories: Graph Coloring Techniques; Heuristic Approaches; and Mathematical Programming Techniques. The following sections discuss each of these categories of approaches in turn.

III.B. Graph Coloring Techniques.

de Werra (1985) discusses the application of graph coloring techniques to the timetabling problem and provides a fairly comprehensive bibliography. He also reports examples of some successful experimental results. However, while these techniques may be promising, the method is beyond the scope of this paper and will not be discussed further. The interested reader is referred to de Werra's article and the bibliography contained therein.

III.C. Heuristic Approaches.

Many different heuristics have been designed to solve specific timetabling problems. The approaches vary greatly in terms of complexity, generally as a function of the specific problem addressed. I will discuss three actual examples of heuristics that have been reported and then make some concluding comments regarding their use. However, to anticipate that discussion, I will state my three chief conclusions up front:

1. Heuristics are generally designed for specific problems. Hence, it may be difficult to generalize a given heuristic approach to other situations.

2. Heuristics are usually implemented on a stand-alone basis and thus may not benefit from advances in other theories, codes, etc. (Mulvey, 1982).

3. Heuristics may yield solutions that are inferior to those obtained through a more direct optimization approach. For example, Tripathy (1980) applied mathematical programming techniques to a problem reported by Barham and Westwood (1978). Tripathy's results represented a significant improvement over those obtained by the original authors who used a heuristic.

The three examples of reported heuristics I will discuss are: Glassey and Mizrach (1986) at the University of California at Berkeley; Barham and Westwood (1978) at Manchester Business School; and Romero (1982) at the Polytechnical University in Madrid. For each example, I will briefly outline the authors' approach and discuss the applicability of that approach to the Sloan School problem.

III.C.1. <u>Glassey and Mizrach (1986)</u>

Glassey and Mizrach (G&M) were faced with a huge timetabling problem at the University of California at Berkeley. They were attempting to assign some 4000 classes to 250 rooms, given meeting times for the classes provided by 80 departments. G&M assumed that half of the rooms could be feasible for each course. Thus, a direct formuation as an integer program would have required approximately (4000 * 250 * 1/2) or a half million binary variables. (Each variable x_{ij} equals one if course i is assigned to room j; it equals zero otherwise.) G&M further estimated that the problem would have approximately 25,000 constraints. Apparently, decomposition by department or day/time was not possible due to the existence of "non-standard" time requests, restrictions on the number of courses each department was allowed to have in "prime time", and other policy considerations.

This problem is clearly too large to be solved directly as an integer program using, say, branch and bound. Moreover, since departments submit their time requests independently, a feasible solution may not even exist to the problem in any given semester. Thus, G&M aimed to develop a heuristic approach which would allow the Berkeley administration to:

1. Quickly determine which time slots would be infeasible due to insufficient room capacity (so that the

affected departments could modify their requests); and

2. Get a rough first pass at a schedule for rooms that would account for most (if not all) of the courses. (If necessary, any "left-over" courses could be dealt with afterward on a manual basis.)

G&M developed a function to assign a "cost" to each of the binary variables. The function included three types of costs: the professor's walking distance from the department's home office to the classroom; a penalty for underutilized facilities (such as assigning a class to a room with video capabilities when that class does not need such facilities); and, a penalty for empty seats. These costs were standardized according to the policymakers' utility functions. (E.g., "y" empty seats were determined to cost the same as a walking distance of "z" yards, etc.)

G&M acknowledge the problem of trading off these multiple objectives but, for lack of any better approach, have designed their heuristic to minimize the total cost of the schedule. The heuristic's strategy is to "solve the hardest remaining problem next". First, the model determines the time slot with the highest (remaining) demand/supply ratio. Within that time period, it iterates around a "greedy algorithm" which assigns classes to the lowest cost room available. The iterations allow for three-way interchanges which would decrease total cost. Finally, after completing the current time period, the heuristic starts over with the next period, etc.

Apparently, G&M's model has performed very successfully. Run time is reported to be approximately one minute of CPU on an IBM 4341 and the Berkeley administration confirms that its solutions are superior to those previously provided by the manual system. Furthermore, the fast turnaround enables the administration to use the system as a Decision Support System which allows for different scenarios as opposed to a "Black Box" which provides one optimal solution. These scenario runs enable the administration to get comfortable with the solutions by determining the key parameters, understanding the sensitivities, etc. Rapid turnaround also helps the administration to modify schedules more easily when various inputs such as course offerings change.

While G&M's heuristic has performed successfully at Berkeley, the key problem in applying their approach to Sloan relates to the issue of resource constraints. As discussed in Section II, at Sloan such constraints include not only rooms, but also requirements that certain courses not be given concurrently (due to faculty, CORE section, or concentration overlaps), etc. At Berkeley, such constraints have presumably been accounted for by delegating the class/time decisions to the departments. (G&M's model only assigns classes to rooms given requested times). Certainly at Sloan these constraints could be accounted for by including them in an extended cost function. However, figuring the appropriate weights for the cost function is likely to be just as difficult as including them more directly as constraints to the schedule. (In fact, calculating the optimal weights is equivalent to using the Lagrangian Relaxation technique discussed later.) Now, given Sloan's size and physical layout relative to Berkeley's, once class times are determined by accounting for these extra constraints, the pure assignment aspect of the problem (i.e., courses to rooms) is fairly simple. Thus, one major goal of a new model at Sloan is to account for these additional issues (as discussed in Section II) more formally.

In summary, Berkeley's decentralized process is very comparable to the current Sloan process. In a four year undergraduate program, students are likely to have sufficient time to take the courses they want even if there are conflicts in any one semester. Thus, there is little need for coordination across departments except with regard to room availability constraints. Hence, G&M's cost function approach is appropriate for Berkeley's problem. At Sloan however, students spend only two years at the school, one of which is dedicated mainly to required courses. Therefore, coordination across departments to prevent potential student conflicts is far more important. In addition, Sloan's smaller size should enable the school to perform such coordination more directly. (The entire Sloan School is probably equal in size to one of Berkeley's larger departments). Thus, as will be discussed in Section IV, I have chosen to model the cross-departmental considerations more directly by way of formal constraints in developing the Sloan School prototype. However, the inclusion of these constraints requires me to use a more complex solution procedure than G&M's heuristic.

III.C.2. Barham and Westwood (1978)

If G&M had the luxury of ignoring some of the cross-departmental issues that we face at Sloan, Barham and Westwood (B&W) faced a problem at the other extreme end of the spectrum. Their problem was to schedule the ten week management program at the Manchester Business School so that participants could take as many of the courses ("options") offered as desired. Of course, this goal could be reached by scheduling every course at a different time. However, the 10 week time constraint on the total length of the program required B&W to formulate the problem as that of finding the minimum number of time periods per week required so that each student could take his top four ("primary") options without any conflicts.

B&W developed a heuristic to group courses together based on the students' preferences. In particular, the groups would be formed such that no student would have included more than one course per group in his four primary selections. In this way, all of the courses within each group could be scheduled at the same time. The heuristic's strategy is to find the most difficult remaining course (i.e., the one with the most conflicts) and then group as many others with it as possible. B&W experimented with various computational techniques to implement that strategy.

The applicability of B&W's approach to Sloan is limited for two reasons. First, the Manchester program has only 36 students and 25 courses. Hence, it is conceivable that many of the course pairs would have no student conflicts at all. At Sloan, there are approximately 200 students and 60-75 electives offered per semester; the probability that there would be no students desiring to take both of any pair of courses is far less likely. My prototype will analyze projected student overlap for specific course pairs and attempt to schedule courses so as to avoid major conflicts. However, this feature is not the only factor used to determine course schedules. In particular, the second shortcoming of B&W's model relative to Sloan is that it does not seem to account for faculty preferences at all. Presumably, the time constraints inherent in a ten week program require the faculty to accept any schedule assigned to them. However, such a process at Sloan is virtually inconceivable since, as discussed in Section II, faculty preferences are required to be a major input into any new scheduling system.

III.C.3. Romero (1982)

Romero reports on a computer support system developed to aid the group decision-making process used to schedule exams at the Polytechnical University in Madrid. (The problem of designing a timetable for examination scheduling is in many ways comparable to the course scheduling problem.) The decision-making group consists of representatives from three constituencies that may have conflicting objectives: the administration, who wants no conflicts in exams; the departments, who want to schedule the exams so as to coincide with the teaching of the subjects; and the students, who prefer that their exams be spread out and who may be concerned with the particular order of exams.

I have included this example to show that a heuristic does not have to be complex in order to be useful. In particular, Romero's model serves mainly as an information system. Exams for particular courses are introduced in sequence and the representatives of the various groups negotiate for an appropriate schedule. The computer helps the negotiators keep track of their objectives and also ensures that the schedules are feasible in terms of room availability, etc. However, the algorithms presented by Romero apparently do not schedule the exams. That task is the responsibility of the negotiators.

Romero claims that the system has helped the decision makers reach adequate agreements more easily. Certainly, any system that can improve the group decision making process is worthy of interest. However, at Sloan (or any other school), the problem of appointing representatives could be difficult given the very different preferences of individuals within the various groups. Moreover, even if such representatives could be appointed, it seems unlikely that they could come to full agreement. In the event of conflict, there would have to be some means of prioritizing their conflicting desires. Romero does not specifically discuss how such potential conflicts would be handled at the Polytechnical University.

<u>Summary of Discussion of Heuristic Approaches</u>

Silver, Vidal, and de Werra (1980) provide a general discussion on the use of heuristics. In the case of timetabling applications, the discussion above presents three examples of very different heuristics that were reportedly successful in addressing specific problems. The advantages of using heuristics are that they can be tailored to specific problems and that they are usually based on intuitive and hence easily understandable techniques. However, as I pointed out at the beginning of this discussion, there may also be disadvantages in using heuristics. To reiterate, such disadvantages may include:

1. Heuristics are problem specific. We have seen how each of the examples I presented might be difficult to apply to the Sloan problem.

2. Heuristics are often implemented on a stand-alone basis and hence may not benefit from new theories, techniques, etc. This would likely be the case for all of the examples I presented.

3. Heuristics may provide solutions that are inferior to more direct optimization techniques. As we will see,

Tripathy (1980) was able to apply optimization techniques to improve on B&W's heuristic. In addition, neither G&M nor Romero have benchmarked their solutions against some (perhaps theoretical) optimum. Thus, it may be possible that their systems are missing some potentially superior solutions without their knowledge.

III.D. Mathematical Programming Techniques

A variety of mathematical programming techniques have been proposed for solving timetabling problems. The "brute force" approach would be to formulate the problem presented at the beginning of this Section as a binary integer program and solve it directly using branch and bound techniques. However, for any timetabling problem of reasonable size, the integer programming method is likely to be computationally impractical. One could solve the linear relaxation of the integer program using the simplex method. Since the simplex method finds corner solutions, one may hope that the linear solution will be integer or that the linear solution may be (more or less) easily rounded. Unfortunately, there is no guarantee that this rounding procedure can be accomplished, without seriously compromising optimality, in a manner any more efficient than that of using formal branch and bound techniques.

In this section, I will describe three different mathematical programming techniques that have been proposed and implemented (to various extents) to solve timetabling problems. These techniques are: (1) A modified linear programming algorithm described by Akkoyunlu (1973); (2) The partial formulation of the timetabling problem as an assignment network with a flexible man-machine interface employed to help the user account for potential non-network constraints (Dyer and Mulvey (1976) and Mulvey (1982)); and (3) Lagrangian Relaxation (Tripathy (1980, 1984)).

III.D.1. Modified Linear Programming.

Akkoyunlu (1973) proposes and implements a modified simplex approach to solving the timetabling problem for one department at SUNY at Stony Brook. He fully specifies the binary integer program as described in Section II, above, including constraints on course conflicts, back-to-back teaching, other administrative policies, and so on. He then applies a modified simplex approach where the key modification is that whenever the potential pivot element is not equal to one, that column is rejected as a candidate for pivoting. In an unpublished paper (Akkoyunlu, 1971), that author shows that this modified procedure is equivalent to applying the simplex method to a feasible region which includes only the (0,1) vertices. The method has apparently been successful; Akkoyunlu reports that a problem with 200 variables was solved in approximately 130 pivots, 50 to get to a feasible solution, and 80 more to obtain the optimal binary solution.

By his own admission, Akkoyunlu's model is realistic with one key exception: he ignores room constraints, claiming that "at least for American universities, this causes no real problem since the classrooms are generally allocated from a campus-wide pool." (Akkoyunlu (1973), p. 347). Thus, while he was able to achieve satisfactory results, it is not clear how much this assumption helped his method obtain a feasible solution for that specific problem. In particular, room availability constraints do exist at the Sloan School.

In addition, Akkoyunlu does not use standard time slots: his model only seeks to find the appropriate number of hours for each class, even if those hours are not the same across days of the week. Clearly, the addition of room availability constraints and the requirement that courses meet at the same time on different days (standard time slots) decrease the degrees of freedom available to a scheduler at the Sloan School relative to one at Akkoyunlu's department. Thus, while his procedure was able to obtain a feasible solution to his problem, it is not obvious that the method would perform equally as well in the Sloan School situation.

III.D.2. Partially Formulate the Timetabling Problem as an Assignment Problem.

Dyer and Mulvey (1976) and Mulvey (1982) developed a timetabling system for the UCLA Graduate School of Management. (Henceforth, these works will be referred to as "Dyer and Mulvey" except where noted.) Their approach had two key aspects. First, they partially formulated the problem as an assignment network. Second, they developed a sophisticated man-machine interface to allow the user to account for the non-network resource constraints in the problem. In terms of the model formulation presented in Section III.A, Dyer and Mulvey explicitly formulated only the network constraints (2) and (3).

The user can develop a schedule iteratively in the following way. The network is solved and the scheduler determines certain critical arcs and nodes whose relationship must be specified to meet certain of the non-network constraints. Such a specification determines a partial schedule which serves as the starting point for the next iteration. The user continues until he is satisfied that all of the non-network constraints are satisfied or that any violations of these constraints are acceptable.

We may note several important points about Dyer and Mulvey's approach. First of all, I have shown at the start of Section III how the timetabling problem can be viewed as an assignment problem with additional complicating resource constraints. Dyer and Mulvey's insight enabled them to design a solution technique based on the integrality property of network problems. This property states that a linear programming solution to a network problem is guaranteed to be integer if all of the coefficients in the problem are integer. (e.g., Bradley, Hax, and Magnanti (1977), p. 344). In particular, in the general formulation of the timetabling problem above, equations (2) and (3) define a unimodular matrix which, if feasible, will yield an integer solution from the application of the simplex method. Since Dyer and Mulvey's formulation does not explicitly include the complicating side constraints, their solution technique will always yield an integer solution. However, it becomes the user's responsibility to ensure feasibility of the schedule with respect to those side constraints.

Second, very large network problems can be generated, stored, and solved very easily, more so even than arbitrary linear programs. In particular, each iteration in Dyer and

Mulvey's system can be solved quickly and easily. In addition, networks can be represented in an intuitive manner using a graphics interface. Dyer and Mulvey claim that such a representation can help the user to understand and work with his problem more easily. This feature is particularly important in their approach since, as stated above, the user is responsible for ensuring feasibility of all of the non-network constraints (such as faculty and student section conflicts, etc.)

Finally, the system is designed to help the scheduler find a satisfactory solution to his problem, not necessarily an optimum. The user can game with various scenarios to explore alternatives, but he can end the process when he is satisfied with the results. The system includes a flexible report writer for outputting the solution in a convenient format.

Dyer and Mulvey's work represents a major contribution to the solution of the timetabling problem. In my work, I have borrowed some of their general concepts as well as specific ideas. As I will discuss in Section V, the prototype method I adopted is also based on an iterative concept. In addition, I have followed their lead with respect to certain specific issues, such as the grouping rooms of like size and the use of an additive choice rule for summing faculty satisfaction ratings. However, in my view the chief potential problem with their approach is that it may be overly reliant on the user to enforce schedule feasibility, particularly as the timetable grows in size and complexity. In Section V, I will discuss how and why I believe my approach is more appropriate for the specific problem at the Sloan School.

III.D.3. Lagrangian Relaxation Techniques

Lagrangian relaxation techniques have recently been applied to a variety of integer programming applications. Fisher (1981, 1985) provide a general survey of Lagrangian Relaxation theory and applications. In particular, Fisher (1985) is a readily understandable treatment of the subject. Somewhat more advanced treatments of the technique and the underlying theory can be found in Fisher, Northrup, and Shapiro (1975), Geoffrion (1974), Shapiro (1971), Brooks and Geoffrion (1966), and the bibliographies contained therein.

Lagrangian relaxation is based on the premise that many difficult integer programming problems can be modeled as a fairly simple problem complicated by side constraints. The goal of the technique is to find optimal penalties

(Lagrangian multipliers) for the complicating side constraints and then "dualize" them by putting them in the objective function. By assumption, the remaining relaxed problem is easier to solve than the original. Typically, the technique involves an iteration scheme where the multipliers are adjusted by a subgradient method (Held, Wolfe, and Crowder (1974)). Unfortunately, while Held, Wolfe, and Crowder do state conditions sufficient for convergence of the subgradient method, such convergence may be very slow, and the objective values may not converge monotonically to the optimal Lagrangian solution.

The chief advantage of the technique is that the tightest upper (lower) bound on the objective obtained by Lagrangian Relaxation for a max (min) problem is guaranteed to be at least as tight as that obtained by a linear programming (LP) relaxation. (Geoffrion, 1974). In fact, the bound obtained by Lagrangian Relaxation will be better than the LP bound in all cases except one. That is the case in which the optimal linear solution to the original problem is exactly integer so that the integrality constraints are not binding anyway. In practice, the iterations of Lagrangian Relaxation often fail to produce an optimal (or even feasible) solution to the original problem. However, Geoffrion's result means that Lagrangian Relaxation can provide a tight bound for an eventual switch to traditional branch and bound techniques. Alternatively, the Lagrangian solution may be nearly feasible and made completely feasible with certain minor modifications. (Fisher, 1985).

Computational experience with Lagrangian Relaxation is reported by Fisher (1973, 1985), and Fisher, Northrup, and Shapiro (1975). Fisher (1972, 1973) deal with the application of Lagrangian Relaxation to scheduling problems. However, Tripathy (1980, 1984) report on the successful application of Lagrangian relaxation to a timetabling problem based on the data from Barham and Westwood (1978). This problem is similar to the one at the Sloan School.

Tripathy's approach is based on the same insight used by Dyer and Mulvey as discussed above. That is, the timetabling problem can be viewed as a pure assignment problem which is complicated by additional side resource constraints. In Tripathy's papers, these side constraints were restrictions on courses that could not be offered at the same time due to student conflicts. Tripathy saw that the problem could be viewed as a classic application of Lagrangian Relaxation by dualizing the complicating constraints. On each iteration the resulting assignment problem was solved and the multipliers adjusted for the next one. Tripathy used the subgradient method presented by

Held, Wolfe, and Crowder (1974) for a maximum of ten iterations and then switched to branch and bound. The branch and bound iterations started with the tightest objective bound obtained by the Lagrangian Relaxation and used the best feasible solution found thus far (if any) as the starting incumbent.

Detailed computational results are presented in Tripathy (1984). To summarize that presentation, Tripathy was able to solve to optimality a fairly large problem (3384 total variables, 1188 dualized constraints, and 275 nodes and 3658 arcs in the underlying network problem) in less than 8 minutes of CPU time on a CDC 7600, Cyber 72, under scope 2.1.4. It should also be noted that Tripathy's original work (1980) was based on the actual problem reported by Barham and Westwood (1978) in which those authors used a heuristic approach. Tripathy's results using Lagrangian Relaxation on the same problem represented a significant improvement over those obtained by the original authors' heuristic.

Summary of Mathematical Programming Techniques

The results obtained in the studies discussed above suggest that there is significant potential for the application of mathematical programming techniques to the timetabling problem. In particular, Tripathy's papers show how Lagrangian Relaxation could be used as a "black box" system while Dyer and Mulvey show how optimization techniques could be applied in a more interactive decision support environment.

In the next section (IV), I will lay out a detailed formulation of the Sloan School timetabling problem. In Section V, I will discuss my own approach for using mathematical programming techniques for the Sloan School timetabling problem, borrowing ideas and approaches from these authors' works.

IV. The Detailed Formulation of the Sloan School Timetable

This section presents the detailed formulation of the Sloan School Timetabling problem as a mathematical programming model. Section IV.A presents the mathematical programming details. Section IV.B discusses some issues raised by the formulation.

IV.A. The Mathematical Programming Model.

As discussed in Section II, the Sloan School defines eight standard time slots during the week. In a typical semester, the school offers approximately 25 CORE courses (including multiple offerings for the different sections) and 60-75 electives during the standard slots. (As discussed in Section II.D, the model's scope excludes courses offered at off hours.) I define variables x_{ij} such that:

$$x_{ij} = \begin{cases} 1 \text{ if course } i \text{ is offered at time } j; \\ 0 \text{ otherwise.} \end{cases}$$

Since there are approximately 100 courses and 8 time slots, there will be about 800 binary variables in the formulation.

The following is a discussion of the objective function and all of the constraints in the model. Each equation will be discussed in detail. A summary of the full formulation appears in Appendix D.

IV.A.1. Objective Function.

The objective function is defined in terms of total faculty satisfaction with the schedule. The objective is to maximize total faculty satisfaction according to an additive choice method for aggregating preferences. (See Section III for a discussion of the problems inherent in additive utility functions.)

Each faculty member was surveyed to obtain his/her input regarding teaching preferences. (See Appendix A for the actual survey.) The faculty were asked, among other things, to rate each of the standard time slots in terms of their preference to teach at that time. The ratings were on a scale of 1 to 5, where 1 is the least desirable rating and 5 is the most desirable rating for a slot. As a first pass, I assumed that the faculty's preferences would be independent of the courses they were teaching. That is, I assumed that if a faculty member liked a given time slot, she would be fairly indifferent as to which specific course she would teach at that time. Analysis of comments returned with the survey data shows that this assumption is not always valid; certain faculty members may have different preferences as to when they might teach a CORE course, say, as opposed to an elective. Strong feelings on this matter are rare, however, and I have decided to stay with the original assumption for now.

The objective is defined as the sum of the utility "contributions" of the course assignments over all i and j:

$$\max \sum_{i} \sum_{j} c_{ij} \, x_{ij}$$

IV.A.2. Constraints.

I have consulted with the Sloan School administration to determine the scheduling policies they would like to see enforced as constraints in the model. Nine types of constraints have been agreed upon. The following is a discussion of each type.

i. Course Assignment.

This constraint merely ensures that each course is assigned to one time slot. To enforce this requirement, we specify that for each course i, the sum of the x_{ij} over all times j equals 1.

$$\sum_{j} x_{ij} = 1 \quad \text{all } i;$$

Since there are approximately 100 courses in the model, there will be approximately 100 of these constraints in the model.

ii. <u>Room Capacity.</u>

Each course has an expected enrollment based on past experience. The expected enrollment determines each course's room size group. Courses at Sloan have been grouped into four room size groups as follows:

Group	Capacity	Number of Available Rooms
R1	capacity >= 90	1
R2	55 <= capacity <= 89	3
R3	25 <= capacity <= 54	6
R4	capacity <= 24	9

While the size cuts for these groupings may seem somewhat arbitrary, the bounds are tighter than they appear. For instance, the three rooms in the second group have an actual capacity of 67-83 students. Thus, the size cuts were made conceding the fact that there may sometimes be a small amount of overcrowding in certain classes in R2. In practice, however, since there is only one room in R1, only the eight largest courses can be offered in that room.

Others must be in a room in R2, regardless of their size. In addition, all of the rooms in the third group have an actual capacity of 40 to 54 students. Thus, there may be some underutilization of certain rooms in R3. However, the reader should note that empirically, room availability constraints tend to be a problem only for the two largest room size groups.

The actual constraint requires that there be enough rooms of each type r available for the courses scheduled at each time j. Thus, if I_r is the set of courses of size group r (r = R1,..,R4) and $N\{r\}_j$ is the number of rooms of type r available at time j, we require that

$$\sum_{i \in I_r} x_{ij} <= N\{r\}_j \qquad \text{for all r and j.}$$

Since $|r| = 4$ and $|j| = 8$, there are 32 of these constraints in the model. ($|r|$ is the "order of r", <u>i.e.</u>, the number of possible values of r.)

iii. <u>CORE Section Feasibility</u>

Each of the twelve CORE sections (A-L) has a set of required courses it must be able to attend without any conflict. In fact, since some of the CORE courses are taught in half-semester format, there are actually two sets of constraints for each section, one for each half-semester. (See the section on formulation issues, IV.B, below, for a more complete discussion of the issues raised by half-semester courses.)

The constraint must ensure that each section receives a feasible schedule. Let I_s^h be the set of courses required for section s in half semester h. Then the constraint is

$$\sum_{i \in I_s^h} x_{ij} <= 1 \qquad \text{for all j, s, and h}$$

Since |j| = 8, |s| = 12, and |h| = 2, there are 192 of these constraints in the model.

iv. Faculty Feasibility

Many of the faculty members teach more than one course during the eight standard periods. Faculty feasibility requires that each such faculty member teach only one course at one time. Thus, let I_f be the set of courses taught by faculty member f. Then the constraint is

$$\sum_{i \in I_f} x_{ij} <= 1 \qquad \text{for all f, j.}$$

In practice, approximately twenty faculty members teach more than one course during peak hours. Thus, there will be approximately 160 of these constraints in the model.

At this time, the reader may recall the discussion at the beginning of Section III. At that time, I pointed out that these four constraints specify the general minimum requirements for a feasible schedule. However, the Sloan School has other policies that the administration would like the model to enforce. These policies result in the addition of the following five groups of constraints.

v. Concentrations.

Sloan School masters' students must fulfill requirements in at least one of approximately fifteen concentrations. The requirements for the concentrations are electives that the students generally take during their second year. This means that the students may only have one or two opportunities to take courses required for their concentrations. Thus, the administration would like to ensure that courses within a concentration be scheduled at different times. This policy aims to give the students a high probability of being able to take those courses required for their concentrations.

Let I_c be the set of courses belonging to concentration c. Then the constraint is

$$\sum_{i \in I_c} x_{ij} <= 1 \qquad \text{for all } j,c.$$

Since $|j| = 8$ and $|c| = 15$, there are 120 of these constraints in the model.

vi. High Conflict Course Pairs.

As discussed in Section II, one chief impetus for the development of a new system was the desire to avoid scheduling courses with a high degree of potential overlap in student demand at the same time. The constraints (v) above will ensure that courses within concentrations not be scheduled concurrently. However, the problem of high-overlap courses across concentrations still remains.

I have addressed the problem as follows. Midway through the current semester, the students were issued a list of the following semester's expected offerings. The students were asked to select their top five elective choices from that list. (See Appendix B for the student survey.) I have written a PASCAL program to compile the survey data and identify pairs of courses with a high degree of potential student overlap. (See Appendix C for the listing and documentation of that program.) Overlap was defined in both absolute as well as relative terms. Absolute overlap within a pair of courses refers to the actual number of students selecting both courses in the pair. Relative overlap is defined as the students who selected <u>both</u> courses in the pair as a percentage of the <u>total</u> number of students who selected <u>either</u> <u>or</u> <u>both</u> courses. The maximum relative overlap for a pair of courses is thus 0.5, in the case where every student selecting either of the two courses selected the other as well.

Relative overlap is important because it adjusts for course size. That is, any two large courses may well have a high degree of absolute overlap purely by virtue of their size. However, the overlap between two smaller courses may be more significant in relative terms to the students in those courses. In my view, both absolute and relative overlap should be considered by the scheduler since both are important measures of potential student conflict between courses. Depending on the results of the survey, the scheduler can decide which course pairs should be scheduled so as to avoid conflict. Section VI discusses an actual test case of the model and describes how the overlap issue was actually handled.

Once the overlap pairs have been identified, constraints can be defined as follows. Suppose there are P such pairs of courses. Let I_p be the two courses in pair p (p = 1,..,P). Then the appropriate constraint is:

$$\sum_{i \in I_p} x_{ij} <= 1 \quad \text{for all } j, p.$$

The number of high conflict pairs p will vary in different years. I expect that there would be no more than 20 such pairs since pairs within concentrations have been accounted for separately. Alternatively, in the interest of obtaining any feasible schedule we might explicitly limit the number of such pairs to, say, twenty. Thus, there will be approximately 160 of these constraints in the model.

vii. <u>Three Hour Seminars</u>

A faculty member may desire to teach a certain course in one continuous three hour session, rather than in two ninety minute slots. If such a course were scheduled wholly within the standard time slots, it would, in effect, consume two time slots (*e.g.*, 9:00 to noon). This would cause two problems. First, students taking the course would have to allocate two of their eight standard periods to it. Second, the course would waste room resources. The latter problem occurs since the use of standard time slots implies that the course's room must be reserved for it on both of a slot's pair of days, even though the course only uses the room once per week. (E.g., if the course met Mondays from 9 to 12, use of standard time slots implies that the room would have to be reserved on Wednesdays from 9 to 12 as well.)

I will discuss this issue in greater detail in the section on formulation issues (IV.B.1) below. For now, it is sufficient to note that, as a first pass, the administration has decided on a policy of scheduling such three hour seminars so as to begin at 2:30 PM. In this way, the second ninety minutes of the session occur after the standard day is over. Since room constraints are not a problem at off times, this approach seems to be a workable solution to the problem. Moreover, if the faculty member and students desire to shift the class to another time, they may be able to do so manually afterwards. In practice, such changes could occur fairly often since these courses tend to be small PhD seminars. As a result, room constraints tend not to be a major problem. In addition, PhD students may have more flexible schedules than masters' students.

The constraint is fairly easy to formulate. Let TSEM be the set of all time slots that do not begin at 2:30, i.e., all slots but Monday/Wednesday and Tuesday/Thursday from 2:30 to 4. Also, let I_{sem} be the set of all such three-hour seminar courses. Then the constraint is

$$\sum_{i \in I_{sem}} \sum_{j \in TSEM} x_{ij} = 0$$

Notice that I have formulated the constraint so that one equation accounts for all of the relevant courses and times.

viii. <u>Each Faculty Member Teaches on a Single Set of Days.</u>

Most faculty members who teach more than one course prefer to teach their courses on only one of the two sets of days. That is, they would prefer to teach all of their courses on either Monday/Wednesday or Tuesday/Thursday. The administration decided to allow the faculty to specify on which of these pairs of days they would prefer to teach. The question was included in the faculty survey (See Appendix A).

As a first pass, the model will attempt to enforce these preferences with constraints. Let FMW and FTT be the sets of faculty members desiring to teach only on Monday/Wednesday and Tuesday/Thursday, respectively. Also, let JMW and JTT be the sets of time slots corresponding to Monday/Wednesday and Tuesday/Thursday respectively. Again, letting I_f be the set of courses taught by faculty member f, we have the following constraints:

$$\sum_{i \in I_f} x_{ij} = 0 \qquad \text{for all } f \in \text{FMW},\ j \in \text{JTT};$$

and

$$\sum_{i \in I_f} x_{ij} = 0 \qquad \text{for all } f \in \text{FTT},\ j \in \text{JMW};$$

The first set of constraints ensures that the faculty members desiring to teach on Monday/Wednesday have no courses on Tuesday/Thursday. The second set of constraints is analogous to the first, but for the faculty members desiring to teach only on Tuesday/Thursday.

I expect that most faculty members teaching more than one course will fall into one of these two groups. Since there are approximately twenty such faculty, and since there are four time slots on each of Monday/Wednesday and Tuesday/Thursday, there will be aproximately 80 of these constraints in the model.

ix. <u>Back to Back Teaching.</u>*

Many faculty members teaching more than one course may, in addition to desiring to teach on only one set of days, also desire to teach (or not teach) back to back. As a first pass, the administration has decided to allow the faculty members to specify this request. Thus, a relevant question was included in the survey in Appendix A. The model will attempt to meet these requests as constraints.

There are four time slots per day: two in the morning and two in the afternoon. The morning and afternoon slots are separated by a one hour lunch period. Thus, for our purposes, back to back has been defined as teaching in both of the AM or PM slots on one set of days. If a faculty member has a lunch break between courses, he is not considered to be teaching back to back.

Now, let

$j = 1$ be the time slot Mon/Wed 9 to 10:30;

$j = 2$ be the time slot Mon/Wed 10:30 to 12;

$j = 3$ be the time slot Mon/Wed 1 to 2:30;

$j = 4$ be the time slot Mon/Wed 2:30 to 4;

---

*I would like to acknowledge the assistance provided by Professor Robert Freund in helping me to formulate the constraints described in this section.

$j = 5$ be the time slot Tue/Thur 9 to 10:30;

$j = 6$ be the time slot Tue/Thur 10:30 to 12;

$j = 7$ be the time slot Tue/Thur 1 to 2:30;

$j = 8$ be the time slot Tue/Thur 2:30 to 4;

Again, suppose I_f is the set of courses taught by faculty member f. There are two cases: (1) f does not want to teach back to back; and (2) f does want to teach back to back. I will consider each of these in turn.

Case (1): Faculty Member does not want to teach back-to-back.

Suppose FMWNBB is the set of faculty members who teach on Mondays and Wednesdays but who do not want to teach back to back. Similarly, let FTTNBB be the analogous set for faculty members teaching on Tuesdays and Thursdays. To ensure that the faculty member does not teach back to back, we merely need to define a constraint which makes sure he/she teaches no more than one course in the morning and no more than one course in the afternoon. (I assume that the maximum load is two courses; very few faculty members teach three or more courses and obviously, if they want to teach all three courses on one day, they will have to teach back to back.)

The constraints for the set FMWNBB are thus:

$$\sum_{i \in I_f} (x_{i1} + x_{i2}) <= 1 \qquad f \in \text{FMWNBB}$$

$$\sum_{i \in I_f} (x_{i3} + x_{i4}) <= 1 \qquad f \in \text{FMWNBB}$$

The first constraint applies to the morning (time periods 1 and 2) and the second to the afternoon (periods 3 and 4). Analogous constraints for FTTNBB are:

$$\sum_{i \in I_f} (x_{i5} + x_{i6}) <= 1 \qquad f \in \text{FTTNBB}$$

$$\sum_{i \in I_f} (x_{i7} + x_{i8}) <= 1 \qquad f \in \text{FTTNBB}$$

<u>Case (2): Faculty Member wants to teach Back to Back.</u>

Now suppose that FMWYBB is the set of faculty members that teach on Mondays and Wednesdays that <u>do</u> wish to teach back to back. Similarly, let FTTYBB be the analogous set for faculty members teaching on Tuesdays and Thursdays. Now, since each faculty member teaches on only one set of days, there are only four possible slots for his two courses. (Again, I exclude the case of faculty members teaching three or more courses.)

To enforce a preference for teaching back to back, we need to include constraints to govern the manner in which the two courses will be allocated to the four slots. In particular, we require that if a morning slot is chosen for one course, then no afternoon slots are chosen for the other and vice-versa. There will thus be four constraints for each set of faculty members (FMWYBB and FTTYBB). Let us consider FMWYBB first. The constraints are:

$$\sum_{i \in I_f} (x_{i1} + x_{i3}) <= 1 \qquad f \in \text{FMWYBB}$$

$$\sum_{i \in I_f} (x_{i1} + x_{i4}) <= 1 \qquad f \in \text{FMWYBB}$$

$$\sum_{i \in I_f} (x_{i2} + x_{i3}) <= 1 \qquad f \in \text{FMWYBB}$$

$$\sum_{i \in I_f} (x_{i2} + x_{i4}) <= 1 \qquad f \in \text{FMWYBB}$$

The first two constraints ensure that if a course is assigned to time period 1 (morning), then no course can be assigned to time periods 3 or 4 (afternoon). The second two constraints are analogous but account for time period 2 in the morning. Similarly, the first and third constraints can be thought of as accounting for period 3 in the afternoon, and the second and fourth account for period 4. The reader may work through the constraints to convince himself that they ensure that the faculty member f will only teach in either the morning or the afternoon.

The constraints for the set FTTYBB are entirely analogous. I state them without explanation except to note that time period 5 replaces time period 1, period 6 replaces period 7, and so on.

$$\sum_{i \in I_f} (x_{i5} + x_{i7}) <= 1 \qquad f \in \text{FTTYBB}$$

$$\sum_{i \in I_f} (x_{i5} + x_{i8}) <= 1 \qquad f \in \text{FTTYBB}$$

$$\sum_{i \in I_f} (x_{i6} + x_{i7}) <= 1 \qquad f \in \text{FTTYBB}$$

$$\sum_{i \in I_f} (x_{i6} + x_{i8}) <= 1 \qquad f \in \text{FTTYBB}$$

It is difficult to estimate precisely how many of these constraints there will be since the exact number will depend on faculty preferences (which may vary). If we assume that half of the faculty desire to teach back to back, and half do not, then if there are 20 faculty members teaching more than one course, there will be approximately 60 of these constraints.

Summary of Detailed Formulation.

Rather than reiterating the formulation here in the text, I have presented a summary of the formulation in Appendix D. That summary includes all of the relevant data sets, parameters, variables, and equations.

The next section discusses some of the issues raised by the formulation. However, before beginning that discussion, I think it is appropriate at this point to stop and take stock. In particular, the reader should note that this formulation is about as complete and realistic as is possible within a mathematical programming framework. All significant constraints have been included. In this way, I believe the model is more complete (or the problem more complex) than many of the examples cited in Section III. For instance, Akkoyunlu (1973) excludes room availability constraints, Mulvey (1982) excludes non-network constraints, Tripathy (1980) and (1984) excludes many of the additional policy constraints (v through ix), and so on.

Of course, the price we pay for reality is increased model complexity. The total number of variables is approximately 800 and the total number of estimated constraints may be approximately 900. Now, many of the constraints can be shown to be interdependent. (For instance, some of the courses within a concentration may be taught by the same faculty member. In such a case, the corresponding faculty feasibility constraints are redundant.) However, the model is large by any measure and is likely to be too large to be solved by "brute force" integer programming techniques.

With this discussion in mind, we can proceed. The next section (IV.B) discusses some issues that should be raised with respect to the formulation. Section V then discusses my solution technique.

IV.B. Issues Raised by the Formulation.

I have stated above that I believe that the model presented in Section IV.A is a very realistic formulation of the actual Sloan School problem. However, I also believe that it is necessary for me to point out six assumptions/simplifications I have made in this formulation and discuss the issues raised by these assumptions. I will discuss each of these in turn.

IV.B.1. Non-Standard Course Formats.

In this section, I refer to "non-standard" courses as those that meet during the standard eight periods but in a "non-standard" format, that is, for a half-semester or in one three hour session. (Non-standard courses that meet on Fridays or on the weekends will be scheduled manually after the model runs are completed.)

The problem with these courses is that they represent a potential waste of resources. Since the vast majority of courses are scheduled in the standard format, a course that meets for a half semester or for only one day will, from the model's perspective, still occupy its full slot. Thus, a course meeting for a half semester will still be allocated its room for the other half, and a course meeting on Mondays (say) will still be allocated its room on Wednesdays.

Furthermore, rooms are only one example of the resources that are wasted in this way. Others include faculty time, student time, etc., all of which must be reserved for the (other) half-semester or day that the course does not actually use. Now, we could correct the problem by scheduling all courses on a half-semester basis, and by individual times (i.e., schedule Monday and Wednesday independently). However, such a solution would greatly complicate the scheduling problem and the resulting model. First of all, splitting the semesters and slots into two parts each implies that the model would require four times as many variables and constraints. Second, we would need to add a whole host of constraints that would serve to try to recreate the standard slots where possible. That is, we would need additional constraints to ensure that if a standard course were scheduled for one session at 9 AM on Mondays (say), then it would be scheduled for its other meeting at 9 on Wednesdays if possible, etc.

Due to the relatively small number of non-standard courses, the added utility of the more complex formulation is likely to be very marginal. Thus, the actual approach I have taken has two aspects:

i. Since most of the half-semester courses are a part of the Masters' CORE, I have added constraints for each half semester to the group of constraints dealing with CORE sections. Thus, the model will "know" that the CORE sections have different required courses in each of the half semesters. The model may thereby schedule courses offered in the different half semesters during the same time slots, if so desired and as long as any other relevant constraints are satisfied.

ii. For all other courses, I will try to manually preprocess the course list to pair up courses across half-semesters or on different days. Thus, if one seminar meets on Tuesdays only and another on Thursdays only, the two seminars in effect can be treated as one course (perhaps with two faculty members). In this way, the courses can use the same room and not waste resources. Of course, in order to be a legitimate pair, the courses need to be roughly the same size (so that they can use the same room) and must have faculty members with compatible teaching preferences. In the case study in Section VI, I will show how this method was effective with one such pair of courses.

IV.B.2. <u>Assignment of Professors to Specific Sections of the Same Course.</u>

All of the CORE courses have multiple offerings. At the present time, some courses have two offerings (each to six of the sections), others have three, four, or six offerings. Since there are twelve CORE sections, the number of sections per offering is, respectively, six, four, three, and two.

The size of the CORE courses is a policy decision made by the faculty and administration prior to the scheduling process. Thus, the model will be given, as input, the number of offerings of each course. However, since the number of sections per offering varies by course, different sections attend different courses together. For example, Microeconomics has two offerings (for Sections A thru F and G thru L respectively) while Accounting has three offerings (Sections A thru D, E thru H, and I thru L) and Strategy has four (Sections A thru C, D thru F, G thru I, and J thru L). Consider section D. For Microeconomics, that section meets with A thru F. For Accounting, it meets with A thru D, and for Strategy it meets with D thru F. The section system at Sloan is thus combinatorially more complicated than if, say, there were twelve sections that attended all of the CORE courses separately from each other.

In Appendix E, I show that the current method of allocating Sections to offerings is in fact combinatorially optimal in terms of minimizing the number of section conflicts between courses. However, an issue still remains as to how to allocate faculty to specific offerings. Since the model assumes that faculty assignments are given, it needs to be provided the faculty member assigned to each of the offerings. For example, it needs to know, say, that Professor Smith is assigned to Strategy A thru C and Professor Jones to Accounting A thru D.

The model takes such assignments as given. However, in the example above, a problem could exist if Smith and Jones have similar time preferences. Since they are both teaching required courses to some of the same sections, they can not teach at the same time. Hence, a better solution could be to assign Professor Brown (say) to teach Strategy to Sections A thru C and let Smith teach Strategy to Sections J thru L (previously assigned to Brown).

In spite of this potential problem, I have continued to preassign the faculty to specific sections for several reasons. First, I could let the model assign the sections to specific professors' offerings endogenously, subject to added constraints that the appropriate number of sections be assigned. While these constraints could be modelled in an integer programming framework, they would add a significant degree of complication and greatly increase the size of the model. For example, since there are twelve sections, I would need twelve additional variables for every CORE course/time combination. I would also need many additional "If/then" constraints. ("If Smith teaches Strategy at time j, then three sections must be assigned to Strategy at that time," etc.)

While the goal of keeping the model less complex is reasonable, we can not afford to sacrifice a significant degree of performance and reality solely for the sake of maintaining simplicity. However, the second reason why I did not add the extra complexity results from an empirical reality: most of the faculty teaching CORE courses teach more than one offering anyway. In some cases, one faculty member may teach the same CORE course to all of the sections (e.g., Microeconomics, Statistics); in others one faculty member may teach two of the three (or four) offerings. (e.g., Accounting, Strategy). Since the faculty member is presumably indifferent as to when he teaches a given course to Sections A thru C versus Sections J thru L, the model is far less restrictive than it may seem at first blush.

Finally, if a faculty member is still very much dissatisfied with his schedule, there is always the possibility of manually exchanging sections with other professors teaching the same course.

IV.B.3. Policies: Constraints vs. Objectives.

As discussed in Section IV.A, above, my model includes as constraints many administrative policies that are beyond the requirements for basic feasibility. (e.g., back-to-back teaching, etc.) There are two problems with this methodology. The first is that by modelling policies as constraints, I have limited the feasible region for the problem and have thereby decreased the probability of finding any feasible solution at all. The second, and perhaps more serious problem, is that the formulation is somewhat inconsistent in that certain preferences are enforced as constraints (back to back) while others (time preferences) are included as objectives. I will consider each of these problems in turn.

i. Limiting the Feasible Region.

In Section II, I described how the current elective scheduling process gives an extraordinary amount of flexibility to the faculty groups and even to individual faculty members. Under this decentralized system, individual faculty members or groups choose the times in which the courses will be offered. Therefore, in many cases back-to-back teaching preferences (e.g.) would be enforced as implicit constraints anyway. Now, while this feature of the current system might have a negative impact on scheduling flexibility, it is a fact of life at the Sloan School. Thus, in order to maximize the probability of the faculty's accepting a new model, I have chosen to "err" on the side of conservatism. That is, I have modelled many of the faculty's preferences as actual constraints. As a result, while it may be more difficult to find a feasible solution to my formulation, <u>any</u> feasible solution that is found is likely to be considered a reasonable first cut by most of the faculty.

ii. <u>Potential Inconsistencies.</u>

In any scheduling system (manual or computer), preferences must be categorized into one of two groups: those that <u>must</u> be enforced versus those that the scheduler would <u>like</u> to enforce. In the Sloan School environment, preferences such as back-to-back, one-day teaching, and so on tend to be more like constraints than are the specific time requests. That is, most (but not all) of the faculty seem to care less about which specific times they teach so long as they teach on one set of days, teach (or not teach) back-to-back, etc. Thus, a model which meets these preferences as constraints but may schedule a course at 9 AM instead of 10:30 is more likely to be accepted than is one in which these other preferences are ignored entirely, in order to keep the model simpler.

It is for these reasons that I view the model as an iterative tool for Decision Support. As I have discussed, the objective function is extremely soft. Since many of the constraints are preferences as opposed to actual requirements for feasibility, the scheduler will need to run various scenarios. Such scenarios could relax certain of the constraints that are enforcing preferences only. These scenarios could thereby test out and compare potential alternatives with respect to the multiple criteria objective that the model is trying to "optimize".

For instance, some of the faculty may want to teach on one set of days, but may be indifferent as to which set they are assigned. Currently, the model requires them (or the scheduler) to choose a set of days anyway. This choice may arbitrarily restrict the range of feasible alternatives. In such a case, I expect that the scheduler would choose the appropriate set of days "cleverly", e.g., by considering preferences of other faculty members teaching courses in the same concentration areas, etc. However, different scenarios certainly should be considered if the first pass solution was perceived to be inequitable or unsatisfactory.

IV.B.4. Faculty Preferences May Differ by Course

My original faculty preference survey (Appendix A) asked the faculty to state their time preferences independent of courses. I assumed that the faculty would be indifferent as to which of their courses they taught at a specific time, as long as the time was acceptable. This assumption, and the fact that many teaching assignments were uncertain at the time of the survey, led me to ask for preferences in the manner described.

In fact, some (but not many) of the faculty do seem to have strong preferences as to which of their courses are taught at what times. (e.g., CORE in the morning, Electives in the afternoon.) The model does have the capability to adjust preferences by specific course. In the future, I would modify the survey to allow the faculty to state their preferences by course, if they so desire.

A more common occurence is for a faculty member's preferences regarding back-to-back teaching to depend on the courses involved. For example, while a faculty member might prefer to teach two offerings of the same CORE course back to back, she might not prefer to teach two different courses back to back. Since in any given semester we know which courses each faculty member will actually teach, such preferences can be properly and directly incorporated into the model.

IV.B.5. Aggregated Preferences.

In Section III, I discussed the Arrow Impossibility Theorem and its implication for group utility functions. At this time, I merely wish to reiterate the fact that the Sloan School model uses an additive choice utility rule, and is thus susceptible to all of the problems inherent in such a rule. I would add, however, that the preference ratings were entirely "free form". Thus, any faculty member could rate only one time slot a 5 and all the others a 1, hoping to get that slot. Other authors have normalized the preference data to prevent such an occurence. For example, I could have asked for a ranking instead of a rating or I could have given each faculty member a fixed number of points which she could allocate in any way she desired.

I decided that such a system would be impractical and unacceptable to the Sloan faculty. In particular, members of the administration and certain key faculty leaders believed that if a faculty member felt strong enough about her preferences to skew them that much, then we would assume that her schedule really was inflexible and try to accomodate her as much as possible. My method obviously opens up the possibility for certain faculty members to game the system, but given the strong sense of community at the school, we did not see this as a problem.

IV.B.6. Room Assignments.

I have made two important simplifications with respect to room assignments. The first is that the model will not assign courses to specific rooms. It will ensure that enough rooms of a given size are available to accomodate the schedule. However, the largely arbitrary task of assigning courses to rooms within the groups will be handled manually after the schedule is set.

Rooms at the Sloan School do not differ much on any dimension other than size. Nearly all rooms have facilities for an overhead projector and most of the larger classrooms have an amphitheatre shape with elevated desks in concentric semicircles. Certain rooms have a single seminar table, but almost all of these are the smaller rooms (capacity less than 25) that the model will allocate to seminar courses anyway. Finally, inasmuch as all of the rooms are physically located within three interconnected buildings, distance between rooms is not an issue.

While some faculty members do have specific room preferences within the size groups, such preferences are generally known to the administration (or can be obtained through a questionaire). Thus, as a first pass, the model will assign courses only to room groups. Assignment of courses to specific rooms within groups will be handled manually after the times are set.

The second simplification I have made is to allocate each course to only one room size group. In particular, larger rooms could accomodate smaller classes if such rooms were available at a given time when all of the smaller rooms were occupied. Thus, the room group constraints could be modelled on a cumulative basis. For instance, courses in room group R3 could fit in any of room groups R1, R2, or R3 as long as there are leftover rooms after scheduling the courses in groups R1 and R2. (Appendix D.2 describes how these cumulative room size constraints could be formulated.)

As a first pass, I have decided to stay with the original formulation for three main reasons. I will discuss each of these in turn.

i. <u>Faculty Preferences.</u>

Most faculty members do not like to teach small courses in large rooms; they would generally prefer to be in as small a room as possible, as long as the students can fit. Inasmuch as this desire is a preference, it is subject to all of the same issues discussed in the section on objectives versus constraints (IV.B.3) above. However, there are two other more compelling factors which justify my original formulation.

ii. Efficient Use of Resources.

Although Sloan is given the first opportunity to choose the rooms in its buildings, other departments at MIT (Economics, Political Science, and others) may schedule courses in rooms left open by Sloan's schedule. To the extent that larger rooms are relatively scarce, the rest of MIT is better off if Sloan schedules its courses to utilize room resources most efficiently. Moreover, if the larger rooms are not used by other departments and are thus kept free, the Sloan administration thereby retains additional flexibility to reschedule courses if actual enrollments exceed expectations. Certainly, if actual enrollment for a given course falls short of expectations, excess room resources may be allocated to it. However, while a small class can always fit in a large room, the converse is obviously not true.

iii. Empirical Realities.

Finally, the empirical facts at Sloan are that the highest demand is generally for the largest rooms. Thus, it is rare that any larger rooms will be free at times when all of the smaller rooms are taken. In particular, there is only one room that can hold more than 83 students and there are plenty of small rooms available for seminars at almost all times.

I do recognize that the alternative cumulative formulation could have merit in suggesting alternative schedules. Thus, I would recommend that the scheduler check every scenario's output to see if there are any time slots in which all of the smaller rooms are occupied but in which larger rooms are available. In such a case, the cumulative formulation can be run as a sensitivity to examine potential alternatives. In fact, in the test case described in Section VI, I do explore the possibilities presented by this sensitivity.

This concludes my discussion of the assumptions and simplifications underlying the formulation as well as their implications for the model's utility. It should be clear that while I acknowledge some of the shortcomings of the model, I feel that it embodies most of the important considerations facing the Sloan School.

The next Section, Section V, discusses my solution technique. Section VI then presents the results of a test run performed for the actual Fall, 1987 semester's schedule.

V. Solution Technique for the Sloan School Timetable.

This section discusses the solution strategy and technique I used to solve the model formulated in Section IV. Section V.A discusses the technique from a conceptual point of view. Section V.B outlines the implementation of a prototype version in GAMS/MINOS on the Sloan School PR1ME 850 computer. Finally, Section V.C provides a very brief discussion of the model's empirical success thus far. Further details are given in Section VI, which describes the actual test runs performed for the Fall, 1987 schedule.

V.A. Conceptual Issues and Discussion.

As I discussed above, the model formulated in Section IV is a fairly realistic but complex depiction of the Sloan School problem. A "brute force" method of branch and bound, say, is likely to be computationally impractical. In addition, no branch and bound code was available to me. I also considered some of the heuristics described in Section III. However, as I discussed at that time, none of those heuristics seemed to be directly applicable to my problem.

My first approach was to apply Lagrangian Relaxation in a manner similar to Tripathy (1980, 1984). That is, I dualized all of the non-network constraints in the formulation. However, I did not find that method to be successful. In particular, runs of up to ten iterations were not successful in yielding a feasible solution to the original problem, or a bound tighter than that obtained by an LP relaxation.

The actual technique I adopted is an iterative approach to Linear Programming. That is, I formulated the full problem as a linear program and relied on a user (me) to perform a "manual but intelligent" branch and bound. I actually adopted this method a bit by accident; when testing Lagrangian Relaxation on an old problem, I found that I could fairly easily modify the results of the first (LP) iteration to obtain an integer solution, generally within 90% or better of the optimum, in less than three iterations. Since the Lagrangian Relaxation technique was failing to yield even a feasible solution within ten iterations, I decided that the LP approach was superior. Conceptually, my technique is similar to Dyer and Mulvey's in that it requires a user to be actively involved in the iterations of the model. However, it is different from theirs in that I have specified the full model and have thereby relieved the user of the responsibility of ensuring feasibility by memory.

While I do not have fixed rules for this manual branch and bound, I can state the two general principles that I found to work well:

1. Use last year's schedule. Last year's schedule is always available for the scheduler to use if the model is having trouble finding a good or feasible schedule. In particular, problems often revolve around the CORE schedule. Last year's CORE provides a very good starting point for this year's problem.

2. Indifferent or Equal Preferences. In some cases where the faculty member had either expressed indifference or had not returned a survey, the model had trouble scheduling the relevant courses. Often it would "split the course in half" between two time slots for which the faculty member was indifferent. In this case, the scheduler can generally help by choosing one or the other. Optimality is not likely to be seriously compromised since if there were other courses requiring that slot, the split course would not have even gotten the fraction of the slot that it did receive.

The next two subsections discuss, respectively, some of the advantages and disadvantages of my technique relative to some of the approaches discussed in Section III.

V.A.1. Advantages.

I believe that the main advantage of my technique lies in the full specification of the model. Again, relative to Dyer and Mulvey, there is far less responsibility placed on the user merely to ensure feasibility of the solution.

Perhaps more significant is the fact that since so many of the preferences have been coded as constraints, any feasible (i.e., binary) solution is likely to be reasonable regardless of optimality as defined by the very soft data incorporated in the objective function. Thus, a formal branch and bound technique to solve the integer program to optimality is probably not necessary for an adequate solution to this problem. We will see in Section VI that analysis of the output schedules must be performed on several dimensions, not just on the basis of one objective value versus another. An iterative approach enables the user to compare various schedules in terms of multiple criteria and objectives.

Second, the model is large, but my personal experience in working with it suggests that it is still sufficiently reasonable in size so that a user can be directly involved in the solution process. The iterative nature of the technique forces the user to know his problem well: understand tradeoffs, sensitivities, key bottlenecks, etc. In turn, this knowledge permits the user to play a valuable role in providing intelligent help to the model by specifying the times for certain courses at various iterations. In particular, since last year's schedule is always known, the user is in a position to help, particularly with respect to the CORE. Moreover, when the schedule is finally published, the user's detailed knowledge of the problem will enable him or her to explain to the faculty why certain decisions were made, which faculty should be asked for alternative preferences, and so on.

Finally, the model will be able to provide relatively rapid feedback to the user as to whether any feasible solution exists at all. In particular, if the linear program is not feasible, then the schedule is not feasible. However, one drawback of the model is that existence of a linear (non-integer) solution does not guarantee the existence of a binary integer solution.

V.A.2. Disadvantages of the Technique.

The primary disadvantage of my technique is that is not guaranteed to find the optimal integer solution. Although upper and lower bounds may be known from previous iterations, we can never be sure that we have an optimum unless the integer value of the objective exactly equals the LP maximum. Thus, the technique requires the user to play an active and intelligent role in the solution process by specifying the times for certain courses during the iterations. In effect, this is the manual aspect of the branch and bound procedure. Now, the user's role can be ambiguous at times. The user cannot specify the times for too many courses all at once lest he use up too many of the model's degrees of freedom and thereby greatly compromise optimality. However, he cannot be overly careful to the point of specifying only one course at a time during the iterations since this would be no better than a "black box" implementation of branch and bound.

The soft nature of the objective function data suggests that strong notions of optimality should not be used. However, a more serious problem may be that even though a feasible linear (non-integer) solution exists, a feasible integer solution may not exist. Thus, the user may continue to iterate along a hopeless path. However, these circumstances could also occur in a traditional implementation of branch and bound. Here, we must rely on the user to provide insight into the problem to end the iterations if he feels they will not lead to a feasible or satisfactory solution.

More generally, the use of <u>any</u> decision support system (DSS) requires the user to first, understand his problem, and second, to understand the capabilities and limitations of the DSS. My prototype shares these characteristics. If used inappropriately, it may well produce poor results. However, if used appropriately, it can provide a significant degree of support to the user in improving the scheduling process and output. Such a case example with actual data will be described in Section VI.

V.B. Implementation of the Prototype.

The linear program was formulated and solved using the GAMS/MINOS system available on the Sloan School's PR1ME 850 computer. GAMS stands for General Algebraic Modelling System. It was developed by David Kendrick and Alexander Meeraus at the World Bank to aid in the development and solution of the mathematical programming models. (Kendrick and Meeraus, (1985)). GAMS is probably among the best of the LP formulation and matrix generation packages available. I consider it to be very "user-friendly", although I would concede that others may find it "user-tolerable". Kendrick and Meeraus (1985) provide a very readable user's guide to the language.

My technique does not in any way depend on a specific software package. However, I chose to use GAMS both because it was readily available and because it was fairly easy and convenient for me to employ. Appendix F provides a listing and documentation of the GAMS formulation actually used in Section VI. Although the documentation is fairly clear, a reading knowledge of GAMS would greatly help to further the reader's understanding of the code. That reading knowledge could be easily obtained by skimming the GAMS manual (Kendrick and Meeraus, 1985). However, a reader experienced with matrix generators and mathematical programming can probably skip that step and read the formulation directly. It should not be particularly difficult for a user to understand the model, if he is familiar with the discussions presented in this thesis document.

GAMS serves only to formulate and generate the model and to write output reports. The actual solution is performed by MINOS, a mathematical programming package. However, MINOS is directly linked to GAMS so that the interface is totally transparent to the user. We need only specify "Solve using LP" in the GAMS formulation and GAMS automatically handles all of the transfer of inputs to and outputs from MINOS. Thus, once the problem is formulated in GAMS, each additional run is trivial to execute.

I would add that no traditional code (PASCAL, FORTRAN, etc.) was required to implement the actual model. This even includes the first few test runs of Lagrangian Relaxation I performed. While GAMS does not have an automatic Lagrangian Relaxation "option", the language is sufficiently flexible so that I was able to model the "dualization" of the constraints and the iterations of the subgradient method wholly within GAMS. (I did write one PASCAL program but its sole purpose was to compile the student survey data.) GAMS includes report-writer capabilites so that reports of the schedule by time, course, concentration, faculty, etc. can be easily implemented.

However, I must emphasize that the implementation is currently at a prototype stage. In particular, I have not yet implemented a complete set of reports, mainly because they were unnecessary for the purposes of this thesis. Output reports have actually been kept to a minimum to save run time and to facilitate my analysis of the outputs. However, more complete reports could be easily designed and implemented in a matter of days.

V.C. <u>Empirical Success.</u>

While the model's empirical success will be discussed in more detail in Section VI, I will briefly summarize some of those results here. Results are based on the test case performed for the actual Fall, 1987 schedule as well as on developmental test runs performed ex post on the Fall, 1986 schedule.

In all cases, I found that, for a given problem, if a feasible LP solution exists, then an integer solution could be found in at most two more iterations. In all such cases, the integer solution had an objective value within 90% or better of the LP optimum. Each run takes approximately 20 minutes on the Sloan School PR1ME 850 computer. This time may vary, generally depending on model size and on how many of the courses are preset for that run. It should be noted that the twenty minutes includes time for model generation as well as the time required to transfer data between GAMS and MINOS. Actual LP solution time is approximately 15 minutes. Finally, certain software experts at Sloan believe that MINOS may not have a particularly efficient implementation of the Simplex method. (It is mainly used for non-linear programming.)

This concludes my conceptual discussion of the model and its prototype implementation. The next section, Section VI, discusses the detailed results of the Fall, 1987 test case. Finally, Section VII provides my conclusions and suggestions for next steps.

VI. A Test Case: The Fall, 1987 Schedule.

The model was originally developed in the early Spring of 1987 using *ex post* data from the Fall, 1986 semester's schedule. Since the model proved to be successful with that data, it was tested more rigorously by being run parallel to the standard scheduling process for the Fall, 1987 schedule. That process began in March of the preceding Spring with a goal of having a tentative schedule sent to the central administration for all of MIT by April.

This section presents and analyzes the model's performance in that actual test case. Section VI.A describes the semester-specific model parameters: courses, faculty, etc. Section VI.B describes the model runs that were performed and the process and criteria I used to choose the best model-generated schedule. Finally, Section VI.C compares that model-generated schedule to the one generated by the existing manual process.

VI.A. Semester-Specific Model Parameters.

VI.A.1. Courses.

A total of 99 courses were planned to be offered for the Fall of 1987. Of these, 12 were scheduled to be offered in the evenings or on Fridays and were thus eliminated from the relevant sample.

Of the 87 remaining courses, two were manually preprocessed so as to be combined into one for scheduling purposes. These courses were two three-hour seminars within the Human Resource Management (HRM) concentration area. In fact, the courses were not actually equivalent in size: one was in room size group R2 while the other was in R4. However, since seminar rooms are always available, the "combined" course was assumed to require only one room (in group R2). The reader may note that in fact, the HRM concentration area had planned to offer three three-hour seminar courses. Thus, unless at least two of those courses were to be paired up, there would have to have been a conflict within the concentration. In addition, the reader may note that since seminar rooms are generally available at all times, the smaller course could be moved by the faculty member and students afterwards.

The 86 net remaining courses consisted of 23 CORE offerings (including multiple offerings of the same courses to different sections) and 63 electives. Fifteen concentrations were represented by the electives with a range of one to seven courses per concentration and an average of three to four courses per concentration. There were ten courses planned to be offered as three hour seminars, four of which were in the CORE.

Room size groupings were a function of expected enrollments based on data from the prior semester. The following table presents a summary of relevant room size grouping data:

Size Group	Number of rooms	Number of slots available	Number of courses	Demand/ Supply Ratio
R1	1	8	8	1.00
R2	3	26[a]	18[b]	0.69
R3	6	48	35	0.73
R4	9	72	23	0.32

Notes:

(a) Includes two extra slots from the Sloan Fellows room;

(b) Net of two joint Sloan Fellows/Masters' courses.

The table can be explained as follows. First, since there are 8 standard time slots per week, in general the number of slots per group is 8 times the number of rooms. Group R2 is the only exception to this rule. There is one additional room in R2 that generally is reserved for the Sloan Fellows. However, that room is available to the other programs for two of the eight periods in the standard week. Thus, the total number of available slots for rooms of type R2 is (3 * 8) + 2 = 26.

Second, two courses in group R2 are joint offerings for the Sloan Fellows as well as the Masters' programs. Those courses are in the same concentration (Operations Management) and the faculty members involved teach only those courses. Since the courses are held in the Sloan Fellows' room, they consume no net resources from the perspective of the Masters' program. Hence, their times have been preset according to the Sloan Fellows' program. In particular, the net number of courses that the model must schedule is thereby decreased by 2 to 84 (8 + 18 + 35 + 23).

Finally, the demand/supply ratios for the groups indicate that R1 is likely be the most difficult room group to schedule since there is no slack in the room resource available to that group. R2 and R3 have roughly comparable demand/supply ratios and, as expected, R4 has the lowest. Thus, R4 (the smallest courses) should be the easiest group to schedule or, if necessary, to move to accomodate conflicts with courses in the other groups.

VI.A.2. Faculty.

The projected offerings for the Fall, 1987 schedule included 55 known faculty members teaching at least one course during the standard hours. Certain courses were known to be taught by new or visiting faculty, but the individual had not yet been determined. The survey return status of the known 55 was as follows:

Status	Number of Faculty	Per cent of Total
Survey Returned	35	63.6 %
MIT faculty but outside Sloan (not surveyed)	3	5.5
Survey Not Returned but Preferences implicitly known and assumed	9	16.4
Survey Not Returned; Preferences Unknown (on leave, out of country, etc.)	8	14.5
Total	55	100.0

MIT faculty members outside of Sloan were not directly involved in the Sloan process. In general, their courses were joint offerings between Sloan and their respective departments and were considered fixed by Sloan. A followup memo and survey (See Appendix A) was sent to those Sloan Faculty members who had not returned surveys. While many did respond to this second memo, we decided not to pursue the remaining non-respondents any further. Nine of these individuals had preferences that were roughly known, and in many (but not all) cases, they were teaching only one small course which could be moved ex post anyway. Finally, there were eight faculty members for whom no preference was known or assumed.

The response rate can also be expressed in terms of courses. The 86 courses can be characterized as follows:

Status	Number of Courses	Per cent of Total
Survey Returned; Preferences known	48	55.8 %
Surveys Returned; Faculty members completely indifferent	5	5.8
Survey Not Returned but Preferences implicitly known and assumed	16	18.6
Survey Not Returned; Preferences Unknown (new or unknown faculty or no survey)	17	19.8
Total	86	100.0

Thus, preferences were known or assumed for the great majority (approximately 80%) of the courses.

Finally, other characteristics of the faculty preferences may be summarized as follows. Twenty-two faculty members were teaching more than one course within the standard week. Of these, 11 desired to teach on Monday/Wednesday, 6 desired Tuesday/Thursday, 3 were teaching Communication to the CORE (spread out), 1 was teaching three courses spread over the week, and 1 was from outside the department but was believed to prefer teaching his courses spread over the week. Four faculty members desired to teach back-to-back (all on Monday/Wednesday) and five desired not to teach back-to-back (two on Monday/Wednesday and three on Tuesday/Thursday). The remaining faculty were indifferent or had not returned surveys.

With one exception, these numbers do not have specific importance beyond determining the overall model size. The original model had 688 (8 * 86) variables and approximately 700 constraints, some of which were interdependent. The one qualitative point that should be noted is the apparent skew in preferences toward Monday/Wednesday slots. The skewed preferences indicate that the schedules will be tighter on those days and that indifferent faculty should probably be assigned to Tuesday/Thursday slots.

This concludes my discussion of the semester-specific model parameters. Section VI.B now discusses and analyzes the actual model runs performed.

VI.B. Model Runs.

Three different scenarios were run for the Fall, 1987 schedule. It turned out that all three scenarios had binary integer solutions to the original linear program so that the manual branch and bound technique discussed in Section V was not needed. However, the reader may recall the empirical discussion of Section V.C. In particular, I continue to believe that the methodology is viable since in other developmental runs, integer solutions were not obtained on the first iteration but were obtained in no more than two additional iterations.

Before proceeding to a detailed discussion of each run, I will present some summary statistics for the three runs.

Run	Description	Number of preset courses	PR1ME 850 CPU minutes	Number of pivots	Optimal Objective Value
1	First Pass	7	20.5	1680	369
2	With Student Survey Data	10	27.0	2176	369
3	Cumulative Room Size Groups	10	27.0	2008	369

All of the runs had 688 variables. Run (1) had approximately 700 constraints. Runs (2) and (3) had 136 additional constraints on seventeen high conflict course pairs (17 pairs * 8 time periods). The run time is moderately long (20+ minutes), but still workable for an iterative approach. Runs were executed in PR1ME's "phantom" mode so the CPU time is approximately equal to run time. While multiple runs may be executed simultaneously, such a practice may not be appropriate, particularly if subsequent runs depend on the output from prior ones. Finally, the reader should note that the optimal objective value obtained from each run was 369. Thus, total faculty satisfaction with each of the scenarios was equal, although there may be different distributions of the same total utility.

The following subsections discuss each of the runs in greater detail.

VI.B.1. <u>The First Run.</u>

<u>Inputs.</u>

The first run was the basic implementation of the model outlined in Section IV with the exception that student overlap data obtained from the surveys was not yet incorporated. All back-to-back, concentration, and other constraints were included. The seven fixed courses consisted of the following:

i. Two joint Sloan Fellows/Masters' courses were fixed, as discussed above.

ii. One elective that was jointly taught by two faculty was strongly desired to be fixed at a given time. The course was in size group R3 so I felt that presetting it would not unduly decrease the degrees of freedom available to the model.

iii. The four offerings of one CORE course (Managerial Behavior) were all required to be assigned to the same time slot. Since those offerings were three-hour seminars, there were only two possible times available. The Sloan School administration previously had agreed with the faculty involved to schedule the four offerings all on Monday (/Wednesday) at 2:30 PM.

Results.

The LP solution of the first iteration of this run (and all the others) was (binary) integer with an overall objective value of 369. In the tables shown above, there were a total of 22 courses for which the faculty member either was completely indifferent (5 courses) or had not returned a survey (17 courses). Since those courses must, by definition, be assigned to a slot for which the preference was indifferent (i.e., rated a 3), the mean satisfaction rating for the remaining 64 courses was (369 - 3(22)) / (86 - 22) or 4.73 out of a maximum of 5. At first blush, this would seem to indicate that the model is performing excellently.

Further analysis confirms this conclusion. In particular, of the 64 courses that had preference data, 61 (95%) were scheduled at times which the faculty member had rated as either a 4 or 5. It is interesting to examine the other three courses individually.

(1). The first course was assigned to a slot that was rated 3. The reason was that the faculty member desired to teach on Mondays and Wednesdays, but there were two other courses within that concentration also scheduled for Mondays and Wednesdays which were assigned to specific slots rated as 5's. The best the model could do was find a slot rated 3 for the remaining course. Note that this course was small (R4). Thus, this faculty member or group could reschedule the course, if desired, by accepting a conflict within the concentration. In fact, one of the two other courses was the one jointly taught course that was fixed as discussed above. Thus, a second option could be to relax the constraint fixing that course. I did not actually test this option, but it is an obvious candidate for a future sensitivity.

(2). The second course was also assigned to a slot rated 3 for similar reasons as #1. The course was in the HRM/IR concentration area which had many course offerings. Again, the course in question was small and could be rescheduled if desired by the faculty member and groups involved.

(3). The third course was actually assigned to a slot rated 1, the lowest rating. However, the reasons were as follows. The faculty member was teaching two courses and desired to teach on Mondays and Wednesdays. He further desired not to teach back-to-back. Thus, he would have to teach one course in the morning and one in the afternoon. However, one of the two afternoon slots was reserved for a seminar course within that concentration. There was thus only one feasible slot in which the afternoon course could be scheduled and the faculty member had rated that slot a 1. The reader may note that I had the opportunity to personally discuss this case with the faculty member involved. When the problem was explained, he agreed that there seemed to be no other solution unless he were to move to Tuesdays and Thursdays, a change which he preferred not to make at this time.

The model was thus able to generate what appeared to be an excellent candidate schedule in one run. However, I had not yet considered the pairs of courses that were high in student conflicts. These pairs were considered in Run 2.

VI.B.2. Student Overlap Considered.

As described in Section IV, the students were surveyed midway through the Spring semester to determine course pairs that were likely to have a high degree of conflict. I received 70 responses from the 185 students in the first year class. (I surveyed only those students that would actually be at the school in the Fall.)

The conflicts were ranked on both absolute and relative terms. The 70 student surveys generated a total of 214 course pairs that had at least one student conflict. In terms of absolute overlap (number of students), these course pairs were distributed as follows:

Number of Reported Student Conflicts	Number of Pairs	Per Cent of Pairs
1	106	49.5 %
2	56	26.2
3	19	8.9
4	14	6.5
5	10	4.7
6 or more	9	4.2
Total	214	100.0 %

In terms of relative overlap (students selecting both courses in the pair as a per cent of total students selecting either or both courses), the distribution was:

Relative Overlap Percentage	Number of Pairs	Per Cent of Pairs
0 - 5%	41	19.1 %
5 - 7.5%	40	18.7
7.5 - 10%	49	22.9
10 - 15%	53	24.8
15% +	31	14.5
Total	214	100.0 %

My first problem was to decide what constitutes a "high overlap" course pair. My first pass decision rule, based on the distributions, was to choose course pairs that had 5 or more overlaps in absolute terms, or 15% in relative terms. Now, many of these conflict pairs consist of courses within the same concentration. Such pairs are already accounted for by the concentration conflict constraints. In addition, there was one popular Finance elective that was to be offered in two sections at different times. I decided not to use any of the pairs including that course since students would have two chances to schedule around it. (It is conceivable that this could leave some students with three-way conflicts around which they could not schedule. However, I have chosen to ignore this possibility.) Finally, many of the top relative overlap courses are also top absolute overlap courses and are thus double-counted in the distributions.

It turned out that after these eliminations, I had 9 course pairs generated from the absolute overlap list and 8 (additional) pairs from the relative overlap list. Thus, there was a total of 17 course pairs for which overlap was to be avoided. Admittedly, my methodology is somewhat arbitrary but it should succeed in catching the most blatant overlap pairs, particularly those which consist of courses in different concentrations.

The schedule generated by the first run had two of the seventeen high overlap course pairs scheduled in conflict. Constraints for the 17 course pairs were thus added to the formulation in the manner described in Section IV.

I would point out that one potential future change in the model could be to eliminate the concentration constraints and deal with all student conflicts in the above manner. For example, if many students concentrating in Finance plan to take more than one Finance elective, that fact would automatically show up in the student surveys. At this point I have chosen to stay with my original formulation, due mainly to the relatively low student response rate. In particular, I believe that there may be more students with conflicts within the concentrations than may appear from my survey data. In addition, students returning their surveys may not yet be aware of the requirements in their concentrations.

Two additional modifications were made to Run 2. First, three additional courses were fixed. Two of these courses were CORE offerings of Statistics. These courses were fixed at the times asssigned to them in the first run. The third course was a Statistics elective which can substitute for the CORE requirement. Nearly all of the students taking that course are first year students substituting it for the CORE course. For this reason, I decided to schedule it at the same time as one of the CORE offerings. In this way, I hoped to minimize the number of students who would have to change sections so as to fit the course into their schedule. The actual slot chosen (of the two) was easily determined since the professor teaching the elective would be new and thus had no known preferences. In addition, the elective course was assigned to the one room in group R1, and the other potential slot was not available since another professor teaching a course in R1 had a very strong preference to teach his course at that time.

The second modification made to this scenario was the movement of one professor from Monday/Wednesday to Tuesday/Thursday. This professor had not returned a survey but was originally thought to prefer Monday/Wednesday slots. However, he was teaching a course that was probably the most difficult to schedule: it was extremely popular, it was one of the courses most often cited in the conflict pairs, and it was taught in R1. A "quick and dirty" analysis of potential changes to the schedule indicated that it would be difficult, if not impossible, to schedule around all of the conflicts unless that course were moved to Tuesday/Thursday. Since the professor had not returned a survey, I assumed that he would not have a strong preference either way and thus would be amenable to the change (although I did not discuss it with him).

<u>Results.</u>

Here again, a binary integer optimum satisfying all of the additional constraints was found in one iteration. The objective value was 369, equal to the value obtained in the first run.

Relative to the solution to the first scenario, 17 of the 86 courses changed slots. All of these 17 moved to slots that had preference ratings equal to those of the slots assigned by the first run. Of the 17, 10 were courses for which there was no survey data, 4 were courses for which the faculty members had claimed total indifference to time slots, and the remaining 3 moved to slots with equal preference (5's) as those assigned by the first scenario.

In my view, this scenario illustrates the power of the model. In particular, it shows how certain courses for which the faculty member has flexible preferences can be rescheduled for the benefit of the entire school. In this case, the improved schedule accomodates additional constraints, based on student input data. Furthermore, it is noteworthy that every faculty member is (apparently) exactly as well off as he was in the first scenario.

VI.B.3. Sensitivity: Cumulative Room Size Groups.

Scenario 3 was formulated exactly the same as #2 with one exception. That exception was that the room size groups were considered to be cumulative: a course in group R3 could be assigned to a room in R2, etc. (The pros and cons of such a formulation were discussed in Section IV.B.6 above.)

The results can be summarized very briefly. A binary integer optimum was again found on the first iteration. The optimal objective value was 369, equal to the values obtained by the first two runs. Seventeen courses were moved relative to Run 2. However, all of these changes were between time slots for which the professor either was indifferent or had not returned a survey. Thus, the results of Scenario 3 do not represent an improvement in total or individual faculty satisfaction relative to those of Scenario 2. It is interesting to note that some of these 17 courses were the same as those moved in Run 2 relative to Run 1. Thus, it is likely that the variables for these courses are non-basic and that the model's choice of specific slots for them may be arbitrary given the lack of strong preferences. (The actual assignment will be a function of the specific pivots performed, which may change when the new constraints are added.) Finally, the most significant result was that there was only one case in which a course was assigned to a room in the next larger group.

In Section IV.B.6 I explained that all else equal, I believe that my original room group forumulation is preferable to the sensitivity. Since the original formulation matches courses to rooms closest in size, it utilizes the room resources available to Sloan and MIT more efficiently than does the sensitivity. For this reason, I have concluded that the schedule produced by Scenario 2 represents the best model-generated schedule.

The next section compares my model-generated scheduled to the schedule generated by the existing process which ran parallel to my effort.

VI.C. Comparison of the Model-Generated Schedule (Run 2) to the Manually Generated Schedule.

I performed a detailed analysis to compare the schedule generated by the model in Scenario 2 to the schedule generated by the standard manual process.

VI.C.1. Faculty Satisfaction.

Relative to the manually-generated schedule, the model-generated schedule had the following results. Of the 86 courses,

- 39 were scheduled in the same slot;

- 16 were moved, but no survey had been returned so no assessment could be made as to which schedule would be preferred by the faculty member;

- 17 were moved, but to slots that were equally preferred by the faculty member according to the survey;

- 8 were moved to slots that represented preference improvements according to the faculty member's survey;

- 6 were moved to slots that were rated lower by the faculty member.

It is interesting to examine each of the last six cases in detail. First of all, four of the six represented moves from slots that the faculty member had rated a 5 to slots that were rated a 4. The other two were each moved from a slot rated 5 to a slot rated 3.

In one case, the faculty member taught two courses and preferred Monday/Wednesday. Due to room constraints (in R1), he could not get his most preferred slot. However, the model was able to assign his two courses to two slots he rated as 4's, the second of which was actually one of the eight improvements the model found. In addition, in the model's schedule, that faculty member teaches all of his courses on Mondays and Wednesdays whereas the manual process had scheduled him to teach one each on Monday/Wednesday and Tuesday/Thursday.

In the three other cases where the courses were moved from slots rated 5 to slots rated 4, the reasons were conflicts with other courses in the concentration, or room constraints (generally R1 and R2). Finally in the two cases where a course was moved from a slot rated 5 to one rated 3, the reasons were also conflicts within the concentrations. However, both of those courses are small, and could be adjusted manually _ex post_ if required/desired by the faculty members involved.

There were other intangible benefits found by the model for three other professors. In one case, a faculty member was successfully assigned to slots on Monday/Wednesday instead of Tuesday/Thursday. Although the faculty member had rated the slots equal on the survey scale, he did state a preference for teaching on Monday/Wednesday. Two other professors were successfully assigned slots all on one day, as opposed to slots spread over the week. In both cases, their surveys had stated this preference.

I did find one problem with the model's schedule. One faculty member was teaching three courses. Two were offerings of the same CORE course and one was an elective. Although she was indifferent with respect to specific time slots, she did desire to teach all on one day, with the two CORE offerings back-to-back either in the morning or afternoon, and with the elective in one of the other free slots. Since she was teaching three courses, the model did not specify back-to-back constraints for her and the resulting schedule assigned her to teach the elective and one of the CORE offerings back-to-back. However, as it turned out, I was simply able to exchange the elective and the other CORE assignment manually and still meet all CORE section, concentration, conflict, and room constraints. With this last change added, I am completely comfortable with the schedule generated by the model.

Finally, the reader may recall the case of the one faculty member assigned to a slot rated 1, discussed above. In fact, the manual process ran into the same constraints as the model. Consequently, he was assigned to that same slot by the manual process as well as by the model.

VI.C.2. <u>Other Means of Comparison.</u>

The reader may note that the schedule generated by the manual process would not be found feasible by the model. Various constraints relating to concentrations, room capacity (the manually generated schedule is apparently allowing for overcrowding), and other teaching preferences (same day, back-to-back, etc.) have been violated.

In addition, the model-generated schedule accounts for all of the top twenty course pair conflicts identified from the student survey data. The manually generated schedule has two of the twenty pairs scheduled in conflict. However, in both of those cases, at least one of the courses in the pair is offered in both the Spring and Fall semesters. Thus, the students should be able to take both during their entire second year.

VI.D. <u>Concluding Comments.</u>

My analysis is not meant to criticize the performance of the current process. I was actually quite surprised to see how well the current process performs, and how relatively few opportunities for improvement actually existed (or were found). Nevertheless, a few apparent improvements were found, thus demonstrating the model's potential for improving the scheduling process.

In summary, I would also reiterate the problems inherent in dealing with relatively soft preference numbers. We must not make the mistake of using the faculty ratings as hard measures of satisfaction. I believe that I have been appropriately conservative in interpreting the preference ratings. When analyzing the results, I have not distinguished between 4's and 5's, and I have not directly traded off one faculty member's improvement in utility with another's decline. Instead, I have made qualitative statements such as "Professor X seems to have experienced an increase in satisfaction while Professor Y is apparently indifferent."

Of course, the reader should recognize that the whole nature of an LP approach implies somewhat difficult comparisons of cardinal utility values across various faculty members. For example, the LP would be indifferent between the following two options: (1) assign two courses to one slot rated 5 and another rated 3; and (2) assign the two courses to two slots rated 4. In contrast, a human scheduler might have a reason (e.g., "equity") to prefer one or the other option. However, the possibility of this type of situation is precisely the reason why I have tried to focus mainly on the proper ordinal characteristics of the preference functions, as opposed to comparisons of the cardinal values in my analyses and evaluations of the alternative schedules. It is also the reason why I have chosen to design a solution technique which requires a human user to be heavily involved in the process.

The next and final section presents some brief conclusions and my suggestions for the next steps required to implement the model formally at Sloan.

VII. <u>Conclusions and Potential Next Steps.</u>

The test case described in Section VI demonstrates that the model is an apparently viable tool for improving the Sloan School scheduling process. The model successfully generated alternative course timetables which appear to be as good as, and generally superior to, timetables produced by the existing manual process.

The model's execution time was 20+ minutes per run. While this may sound like a lot, I would point out that run time is not the only measure of turnaround time. In any case, run time needs to be viewed from the proper perspective. In particular, the reader may note that once the model was developed, and the semester-dependent data collected and input, the scenarios discussed in Section VI were generated all in one day. The data collection, input, and analysis were all performed within two weeks. Thus, the school could consider hiring an undergraduate Research Assistant to work with the model and perform the work described in Section VI on a regular basis each semester.

I have divided my suggested next steps into two sections. The first deals with the model itself and the second addresses the general scheduling process.

VII.A. Model Development.

Two aspects of the model should be further developed. These are its extension for use in scheduling a Spring semester and the further extension of the user interface.

VII.A.1. Extensions Required for Use in Scheduling the Spring Semester.

The model was originally developed for a Fall semester. The issues in a Spring semester are conceptually the same, with two exceptions/additions.

i. Friday Sections of CORE Courses.

Certain CORE courses have additional sessions on Fridays. Some faculty members apparently desire to teach such sessions at the same times on Fridays as they do during the week. Thus, if a course is taught on Monday/Wednesday at 9 AM, faculty preferences suggest that the Friday session should also be at 9. In the fall, this consideration is not an issue since only one course each half-semester meets on Fridays. Thus, that course can be easily scheduled at the same time on Friday as it is during the week.

However, in the Spring, a student may have three or even four courses meeting on Fridays, depending on the student's concentration (discussed below). Thus, in order to permit each faculty member to teach at the same time on Fridays that she does during rest of the week, the CORE schedule must be designed so that no more than one course with a Friday session is scheduled at the same hour across the two sets of days. For example, Finance and Operations both have Friday classes. In order for the professors involved to teach at the same time on Friday as they do during the rest of the week, we must require that if Finance and Operations are taught on different days, they must not be taught at the same time (*e.g.*, 9 AM) to the same sections.

The required constraints can be formulated as follows. Let I_{sf}^{h} be the set of CORE courses requiring a Friday meeting that are taken by section s in half-semester h. Further, let J_t be the set of time slots that occur at the same time, but on different days. Thus,

J_1 = { Mon/Wed 9-10:30; and Tue/Thur 9-10:30 }

J_2 = { Mon/Wed 10:30-12; and Tue/Thur 10:30-12 }

J_3 = { Mon/Wed 1-2:30; and Tue/Thur 1-2:30 }

J_4 = { Mon/Wed 2:30-4; and Tue/Thur 2:30-4 }

The constraints are:

$$\sum_{i \in I_{sf}^{h}} \sum_{j \in J_t} x_{ij} <= 1 \qquad t = 1..4; \quad s = A..L; \quad h = 1,2.$$

It should be noted that the addition of these constrants is not a requirement for schedule feasibility. The decision to include these constraints should be based on faculty preferences.

ii. Mapping of Concentrations to Sections.

The second major difference in the Spring is the fact that first-year students in different concentrations have different requirements. Finance and Marketing (and potentially Operations Management) have full semester CORE requirements for students in those concentrations. Other students take condensed half-semester versions of the same courses.

At present, the sections are composed randomly, with no consideration to given concentrations, etc. Now, this may be an appropriate educational policy since one of the main goals of the school is to encourage student camaraderie, etc. However, in the Spring the policy leads to a very difficult scheduling problem. In essence, the number of de facto sections is multiplied by four: Section A, Finance Concentrators; Section A, Marketing Concentrators; Section A, Marketing and Finance Concentrators; and Section A, All Other Concentrators; etc. This phenomena could be modelled by increasing the number of de facto sections in order to maintain feasibility for all the relevant concentrations. However, such a feasible solution may be difficult to find, and may unduly restrict the alternatives available.

A simpler solution would be to reassign sections at the beginning of the Spring on the basis of concentrations. This would decrease the number of Section/Concentration combinations required to be feasible. Meanwhile, the students would have been given the chance to meet people outside of their concentrations during the Fall semester so that excessive formation of cliques, etc. need not be a problem. It should be noted that, in effect, the manual process is doing this already since the actual Spring, 1987 schedule is not feasible for all conceivable Section/Concentration combinations. Some students must be changing sections (formally or informally) in order to meet their requirements under the present system.

The summary of the model formulation in Appendix D includes a presentation of these additional constraints.

VII.A.2. Further Development of a User Interface.

The second major area of model development lies in the user interface. As I discussed at the end of Section V, the current interface has been designed only to facilitate my development of the prototype and analysis of the results. In the future, the user interface can be extended in three possible directions:

i. Information System (MIS) Aspects.

At a minimum, additional output reports need to be designed within the GAMS model. Such reports would serve two functions: first, to output the results in a more convenient format; and second, to facilitate performance of the analyses presented in Section VI on a more regular basis each semester. These reports could be implemented in a matter of days by a user familiar with GAMS once the required formats were agreed upon. Other MIS extensions could be improved means for data input and storage, perhaps using a relational database system such as SQL. In fact, GAMS stores data in a relational database that could itself be used. (Kendrick and Meeraus, 1985).

ii. <u>Graphic Interface.</u>

A second direction would be to explore the possibility of developing a graphic interface, perhaps based on Dyer and Mulvey's example. One very interesting possibility would be to implement the user interface in an "off-line" environment, such as a Macintosh. Such an implementation could make liberal use of the Mac's graphic capabilities. Unfortunately, the design and implementation of such a system were far beyond the scope of this thesis.

iii. <u>Expert Systems.</u>

One potentially promising application of expert systems could be their use as intelligent interfaces between human users and large-scale mathematical programs. In this case, an expert system could be given various scheduling rules and means for interpreting the output of the LP for the user. Such an expert system could potentially improve performance of the manual branch and bound technique, if it were necessary. It might also be able to choose among alternatives that the LP rated as (apparently) indifferent. Here again, I can only make the general suggestion; a detailed implementation is beyond the scope of my thesis.

VII.B. The Scheduling Process.

I conclude this thesis with some comments and observations about the general scheduling process at Sloan.

Ultimately, successful implementation of my or any other scheduling system will require far more than a model. First, the faculty and administration must decide what they really want from such a system. I have taken my best passes at balancing the interests and preferences of the various faculty members and of the students. However, the final decisions must be made by the faculty and administration, hopefully based on my analysis and suggestions.

Second, the school must decide how far it is willing to go in enforcing the use of the system. In particular, incentives must be given to both the faculty and students to encourage the completion and return of their respective surveys. The faculty response rate was very encouraging. However, I was missing surveys from several key people. Moreover, the faculty who did return surveys may have done so only because the project was viewed as an academic endeavor. The response rate may have been quite different (in either direction) had the faculty been told that their actual schedules would be generated by the model.

In addition, the students' response rate (37%) was somewhat discouraging to me. While it is true that the students, as always, were busy at the time of the survey (see Appendix B), it could not have taken them more than a half hour to complete. Furthermore, the students are the constituency with the most to gain from the model since they have zero input now.

One means of increasing both the student and faculty response rates would be to simply publicize the results of the model. In particular, when the faculty become aware of the overwhelming percentage (95%) of courses which had survey data assigned to slots rated as 4's or 5's, they might be more likely to respond to preference surveys. Similarly, if the students are shown how the system successfully decreased their conflicts, they might also be more inclined to return their surveys.

In conclusion, I would say that the model seems to hold promise for improving the Sloan School scheduling process. More work needs to be done in extending the model and in implementing it within the Sloan culture and context. However, I do believe that my work represents a good start and I recommend that a more formal implementation be undertaken in the near future.

APPENDICES

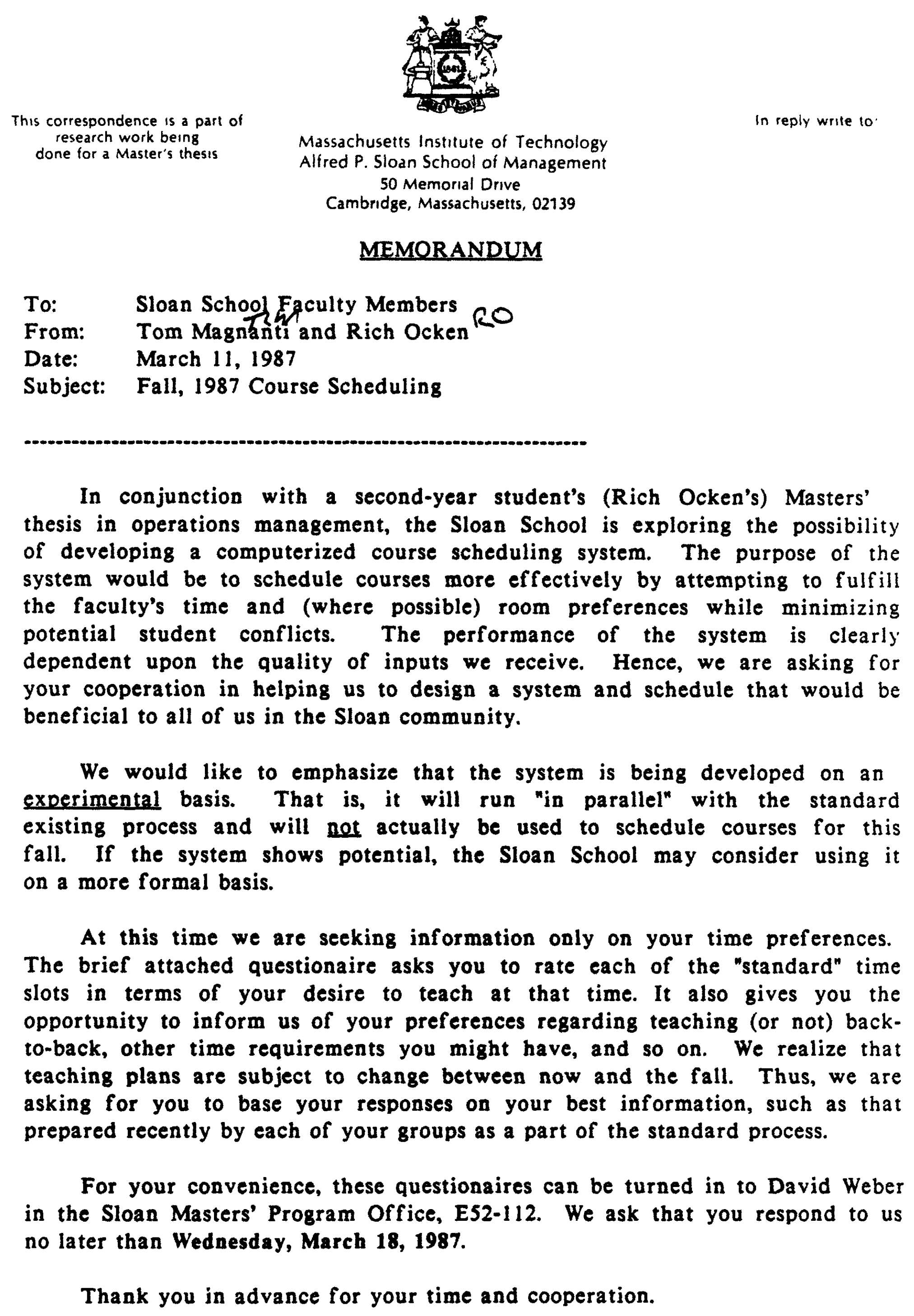
Appendix A: <u>Faculty Survey.</u>

This appendix presents the survey and accompanying memos that were sent to the faculty to determine their preferences. Figure A-1 presents the original cover memo sent with the survey. Figure A-2 presents the survey itself and Figure A-3 presents a suggested revised version of one of the questions. Finally, Figure A-4 presents a followup memo sent to faculty members who did not return their surveys on time.

A.1. <u>The Original Cover Memo.</u>

The survey was originally sent with a cover memo (Figure A-1) from myself and Professor Tom Magnanti, the Area Head of the Sloan School's Management Science Area. The memo was intentionally designed to be as non-threatening as possible. That is, we explained to the faculty that the model would <u>not</u> actually be used to schedule courses this semester. However, the faculty were informed that if the model showed promise, the school might consider implementing it more formally in the future. In this way, we hoped to obtain the maximum amount of faculty cooperation possible.

Figure A-1: Cover Memo sent with the Faculty Survey.



This correspondence is a part of research work being done for a Master's thesis

Massachusetts Institute of Technology
Alfred P. Sloan School of Management
50 Memorial Drive
Cambridge, Massachusetts, 02139

In reply write to:

<u>MEMORANDUM</u>

To:	Sloan School Faculty Members
From:	Tom Magnanti [illegible] and Rich Ocken RO
Date:	March 11, 1987
Subject:	Fall, 1987 Course Scheduling

In conjunction with a second-year student's (Rich Ocken's) Masters' thesis in operations management, the Sloan School is exploring the possibility of developing a computerized course scheduling system. The purpose of the system would be to schedule courses more effectively by attempting to fulfill the faculty's time and (where possible) room preferences while minimizing potential student conflicts. The performance of the system is clearly dependent upon the quality of inputs we receive. Hence, we are asking for your cooperation in helping us to design a system and schedule that would be beneficial to all of us in the Sloan community.

We would like to emphasize that the system is being developed on an <u>experimental</u> basis. That is, it will run "in parallel" with the standard existing process and will <u>not</u> actually be used to schedule courses for this fall. If the system shows potential, the Sloan School may consider using it on a more formal basis.

At this time we are seeking information only on your time preferences. The brief attached questionaire asks you to rate each of the "standard" time slots in terms of your desire to teach at that time. It also gives you the opportunity to inform us of your preferences regarding teaching (or not) back-to-back, other time requirements you might have, and so on. We realize that teaching plans are subject to change between now and the fall. Thus, we are asking for you to base your responses on your best information, such as that prepared recently by each of your groups as a part of the standard process.

For your convenience, these questionaires can be turned in to David Weber in the Sloan Masters' Program Office, E52-112. We ask that you respond to us no later than **Wednesday, March 18, 1987.**

Thank you in advance for your time and cooperation.

A.2. The Survey.

The survey itself (Figure A-2) is fairly straightforward. The faculty were given the opportunity to:

1. Specify courses offered in off-hours; (Q2)
2. Rate the time slots; (Q3)
3. Specify their preferences re:
   - Single day teaching; (Q4a)
   - Back-to-back teaching; (Q4b)
4. Specify other time commitments they might have; (Q5)
5. Make other comments/suggestions as they saw appropriate. (Q6)

There are two specific changes I would make to the survey, in questions (3) and (4b). I will describe each of these in turn.

First, although the survey instructions were fairly clear, there was apparently some confusion with respect to the rating scheme. In particular:

1. Some of the returned surveys indicated that the faculty respondent had rated his most preferred times as 1's

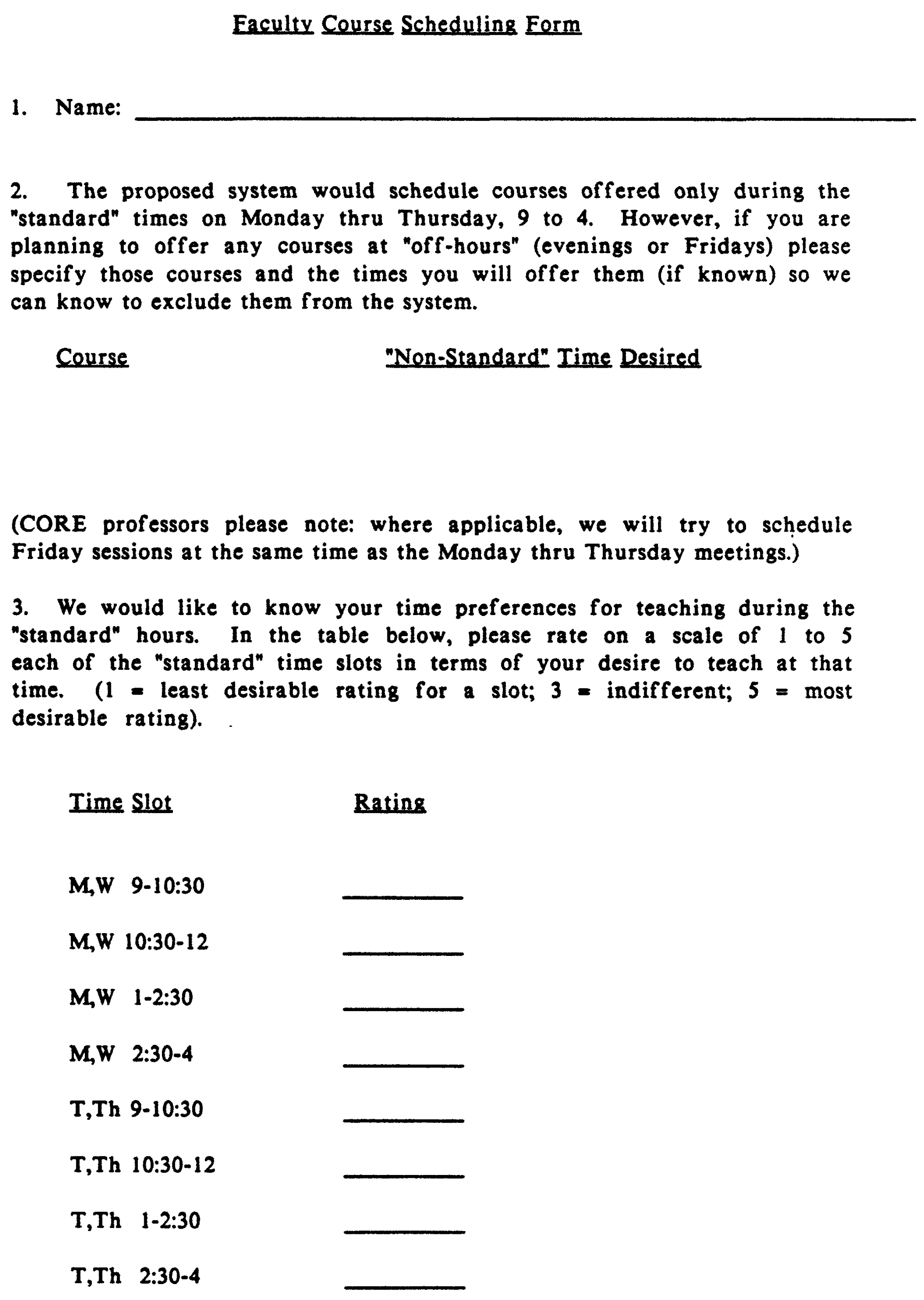
and 2's, as opposed to 5's and 4's. I determined this from some of the comments on the surveys. For instance, one faculty member rated slots on Monday/Wednesday as 1's and 2's, but then stated he preferred to teach on Monday/Wednesday in responding to Question 4a.

2. Some faculty members ranked the slots on a scale of 1 to 8, instead of rating them. As I explained in the text (Section IV.B.5), I chose to allow the ratings to be "free form", i.e., I did not require the faculty member to rank the time slots.

For the test run, I adjusted the (known) incorrect surveys by inverting the ratings where necessary and/or normalizing the rankings on a scale of 1 to 5. However, to avoid confusion in the future, I would redesign Question 3 in the survey as shown in Figure A-3. In particular, by asking the faculty to circle their preference ratings, I would expect to be certain of obtaining proper responses.

The second change pertains to the back-to-back question, Question 4b. To avoid any possible confusion, I would specify clearly that the time slots that are separated by lunchtime (10:30-12 and 1-2:30) are not considered to be back-to-back.

Figure A-2: The Faculty Survey (page 1 of 2).



<u>Faculty Course Scheduling Form</u>

1. Name: ____

2. The proposed system would schedule courses offered only during the "standard" times on Monday thru Thursday, 9 to 4. However, if you are planning to offer any courses at "off-hours" (evenings or Fridays) please specify those courses and the times you will offer them (if known) so we can know to exclude them from the system.

Course	"Non-Standard" Time Desired

(CORE professors please note: where applicable, we will try to schedule Friday sessions at the same time as the Monday thru Thursday meetings.)

3. We would like to know your time preferences for teaching during the "standard" hours. In the table below, please rate on a scale of 1 to 5 each of the "standard" time slots in terms of your desire to teach at that time. (1 = least desirable rating for a slot; 3 = indifferent; 5 = most desirable rating).

Time Slot	Rating
M,W 9-10:30	________
M,W 10:30-12	________
M,W 1-2:30	________
M,W 2:30-4	________
T,Th 9-10:30	________
T,Th 10:30-12	________
T,Th 1-2:30	________
T,Th 2:30-4	________

Figure A-2: The Faculty Survey (page 2 of 2).

4. Please answer questions 4a and 4b if you will be teaching two or more courses during "standard" time slots.

4a. Do you prefer to teach all of your courses on one set of days (i.e., Monday/Wednesday or Tuesday/Thursday)?

Which set of days (Mon/Wed or Tues/Thur) do you prefer?

4b. Do you have a strong preference to teach/not teach back-to-back?

Preferred	________
Indifferent	________
Not Preferred	________

5. Do you have any other time commitments (*e.g.*, Sloan Fellows, Research Seminars, etc.) for which we should reserve free time for you? Please specify.

6. Do you have any other comments and/or preferences you would like us to know about?

Figure A-3: Redesign of Survey Question 3.

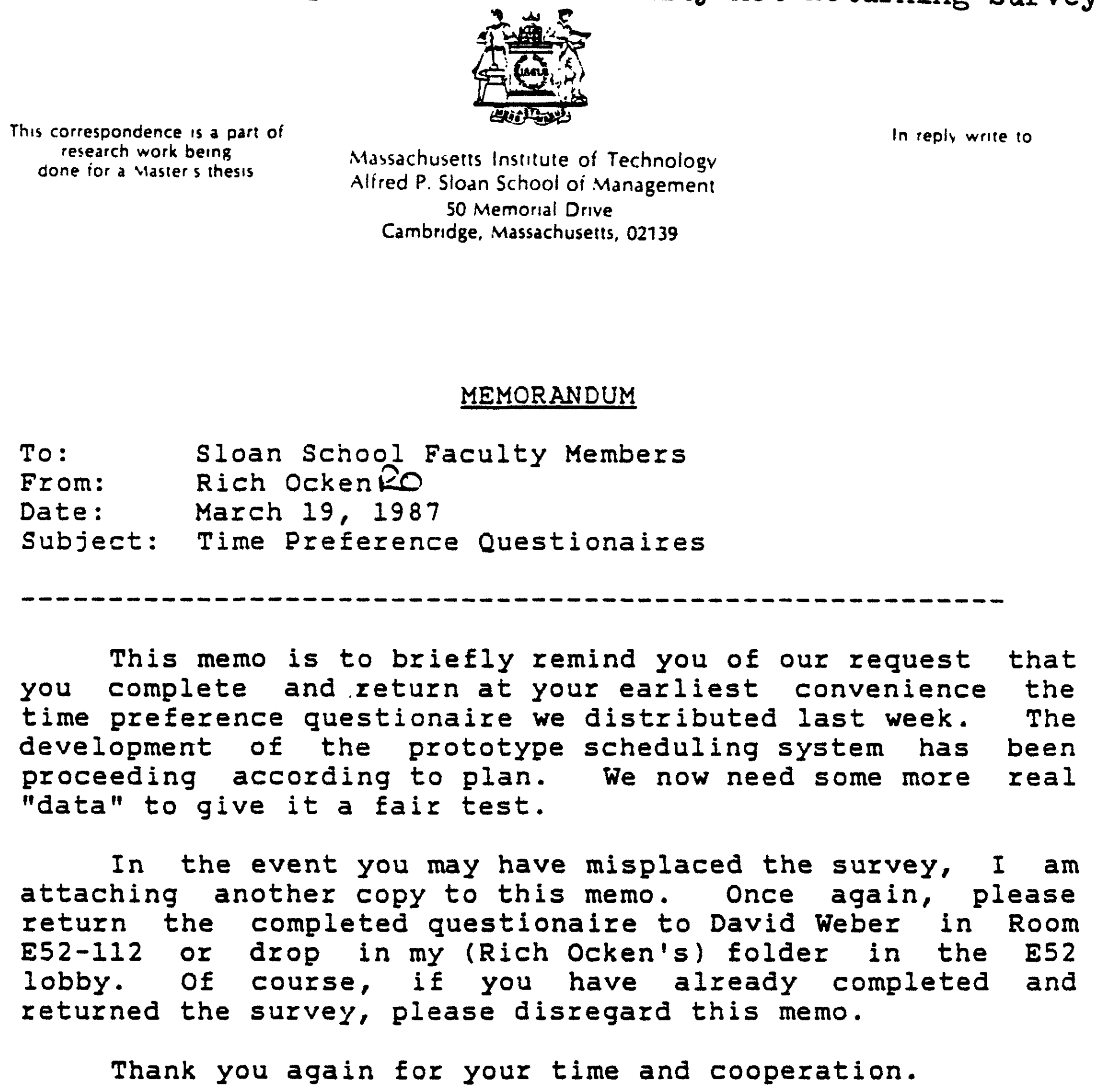
3. We would like to know your time preferences for teaching during the "standard" hours. In the table below, please rate on a scale of 1 to 5 each of the standard time slots in terms of your desire to teach at that time. Please rate each time slot by circling the number corresponding to your preference. (1 = least desirable rating for a slot; 3 = indifferent; 5 = most desirable rating).

Time Slot	Rating				
	Least Desired		Indifferent		Most Desired
M,W 9-10:30	1	2	3	4	5
M,W 10:30-12	1	2	3	4	5
M,W 1-2:30	1	2	3	4	5
M,W 2:30-4	1	2	3	4	5
T,Th 9-10:30	1	2	3	4	5
T,Th 10:30-12	1	2	3	4	5
T,Th 1-2:30	1	2	3	4	5
T,Th 2:30-4	1	2	3	4	5

A.3. <u>The Followup Memo.</u>

Figure A-4 presents the followup memo that was sent to faculty members who did not return their surveys on time. No explicit changes need to be made to the memo itself. However, I would hope that in a formal implementation of the model, more faculty members would return their surveys on time and that this memo would be needed only as a reminder.

Figure A-4: Followup Memo sent to Faculty not Returning Surveys.



This correspondence is a part of
research work being
done for a Master's thesis

Massachusetts Institute of Technology
Alfred P. Sloan School of Management
50 Memorial Drive
Cambridge, Massachusetts, 02139

In reply write to

MEMORANDUM

To:	Sloan School Faculty Members
From:	Rich Ocken RO
Date:	March 19, 1987
Subject:	Time Preference Questionaires

This memo is to briefly remind you of our request that you complete and return at your earliest convenience the time preference questionaire we distributed last week. The development of the prototype scheduling system has been proceeding according to plan. We now need some more real "data" to give it a fair test.

In the event you may have misplaced the survey, I am attaching another copy to this memo. Once again, please return the completed questionaire to David Weber in Room E52-112 or drop in my (Rich Ocken's) folder in the E52 lobby. Of course, if you have already completed and returned the survey, please disregard this memo.

Thank you again for your time and cooperation.

Appendix B: <u>The Student Survey.</u>

Figure B-1 presents the survey sent to the students to determine their expected course preferences. The first page is the actual memo and survey; the next three pages contain the projected course list for the Fall, 1987 semester.

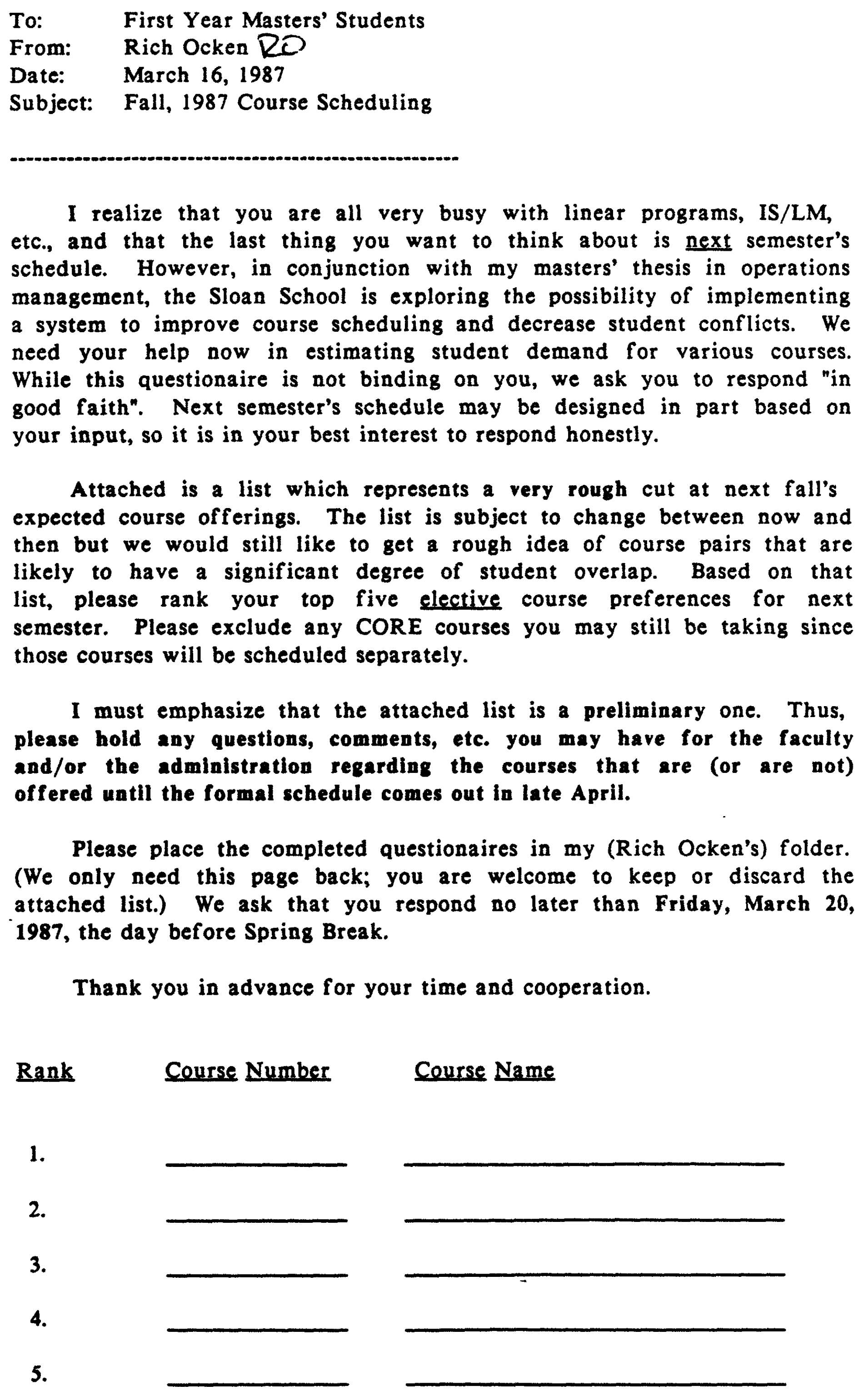
The surveys were individually distributed to each of the first-year students' folders. (All Sloan School students have folders centrally located in the school's lobby.) Since this model was being developed as a prototype, I asked the students to return their surveys to my folder, rather to the school's Masters' Program Office. It is conceivable that the student response rate would have been somewhat higher if the survey had been distributed and collected by the administration. However, this conclusion is not entirely certain since I may have benefited from student camaraderie and empathy.

Timing of the Survey.

There is probably some optimal time for distributing these surveys. Such a time must balance conflicting factors. The survey cannot be distributed very early in the preceding semester for two reasons. First, we need to know the courses that are planned to be offered. Second, the students may not take the survey seriously, or may not know their preferences, if it is distributed very early. However, the survey cannot be distributed too late since we would want to have enough time to allow for changes and iterations after candidate schedules are presented to the faculty.

The survey was actually distributed on the Monday before the school's Spring Break. I felt that a week would be enough time; any students who did not return the surveys after that time would probably never return them. However, in addition to the Spring Break problem, that week is also the week of finals for the two half-semester courses offered in the first half of the Spring semester. These factors may have decreased my yield a bit. Thus, in the future, I would recommend that the surveys be distributed a week earlier.

Figure B-1: The Student Survey (page 1 of 4).



To: First Year Masters' Students
From: Rich Ocken RO
Date: March 16, 1987
Subject: Fall, 1987 Course Scheduling

I realize that you are all very busy with linear programs, IS/LM, etc., and that the last thing you want to think about is next semester's schedule. However, in conjunction with my masters' thesis in operations management, the Sloan School is exploring the possibility of implementing a system to improve course scheduling and decrease student conflicts. We need your help now in estimating student demand for various courses. While this questionaire is not binding on you, we ask you to respond "in good faith". Next semester's schedule may be designed in part based on your input, so it is in your best interest to respond honestly.

Attached is a list which represents a very **rough** cut at next fall's expected course offerings. The list is subject to change between now and then but we would still like to get a rough idea of course pairs that are likely to have a significant degree of student overlap. Based on that list, please rank your top five elective course preferences for next semester. Please exclude any CORE courses you may still be taking since those courses will be scheduled separately.

I must emphasize that the attached list is a **preliminary** one. Thus, **please hold any questions, comments, etc. you may have for the faculty and/or the administration regarding the courses that are (or are not) offered until the formal schedule comes out in late April.**

Please place the completed questionaires in my (Rich Ocken's) folder. (We only need this page back; you are welcome to keep or discard the attached list.) We ask that you respond no later than **Friday, March 20, 1987, the day before Spring Break.**

Thank you in advance for your time and cooperation.

Rank	Course Number	Course Name
1.	________	________________
2.	________	________________
3.	________	________________
4.	________	________________
5.	________	________________

Figure B-1: The Student Survey (page 2 of 4).

Expected Elective Offerings, Fall, 1987

Applied Economics

15.001	Managerial Economics (Undergraduate course)
15.013	Industrial Economics for Strategic Decisions
15.018	Economics of International Business
15.034	Applied Econometrics and Forecasting for Management
15.035	Pricing Strategy (new course)
15.041	Research Seminar in Applied Economics

Operations Research/Statistics

15.053	Introduction to Management Science (undergraduates)
15.058	Applied Mathematical Programming
15.059	Mathematical Programming Models and Applications
15.065	Decision Analysis
15.073J	Introduction to Stochastic Processes
15.075	Applied Statistics
15.078J	Logistical and Transportation Planning Methods
15.081J	Introduction to Mathematical Programming
15.083	Combinatorial Optimization
15.089	Workshop in Operations Research
15.099	Doctoral Statistics

Health Care Management

15.121	Seminar in Health Management
15.141J	Comparative Health Systems
15.149	Special Studies in Health Management

International Management

15.221	International Business Management
15.231	Mgt. and Tech. in People's Republic of China (new)
15.232	The Firm and Business Environment in Japan (new)

Communication

15.281	Advanced Managerial Communication

Figure B-1: The Student Survey (page 3 of 4).

Expected Elective Offerings, Fall, 1987 (continued)

Organization Studies

15.301	Managerial Psychology Laboratory (undergraduate)
15.312	Managerial Decision Making and Leadership
15.317	Comparative Study of Organizations
15.345	Doctoral Seminar in Organization Studies
15.347	Doctoral Seminar in Research Methods I

Management of Technology and Innovation

15.351	Managing Technology and Innovation
15.371	The R&D Process: Communication and Problem Solving
15.965	Special Seminar: Technology/Marketing Interface

Finance

15.412	Financial Management II
15.413	Topics in Corporate Financial Management
15.415	Finance Theory
15.435	Corporate Financing Decisions
15.436	International Managerial Finance
15.437	Options and Futures Markets
15.438	Investment Banking and Markets
15.441	Research Seminar in Finance

Accounting

15.501/516	Financial and Cost Accounting (undergraduate)
15.521	Management Accounting and Control
15.525	Corporate Financial Accounting
15.539	Special Seminar in Accounting, Planning and Control
15.951b	Introductory Managerial Accounting (undergraduate)

Management Information Systems

15.562	Principles of Management Information Systems
15.564	Management Information Technology I
15.565	Management Information Technology II
15.568	Management Information Systems

Figure B-1: The Student Survey (page 4 of 4).

Expected Elective Offerings, Fall, 1987 (contd)

Law

15.601/611	The American Legal System
15.615	Manager's Legal Function: From Birth to Bankruptcy
15.963	Tax Law

Industrial Relations and Human Resource Management

15.664	Management of Human Resources
15.665	Power and Negotiation
15.671J	Labor Economics I
15.674J	Comparative Systems of IR and Human Res. Development
15.691J	Research Seminar in Industrial Relations

Operations Management

15.763	The Practice of Operations Management
15.768	Operations Management in the Service Industry
15.769	Manufacturing Strategy (new course)

Marketing

15.812	Marketing Management
15.824	Marketing Communications
15.825	Marketing Models
15.832	Measurement for Management
15.839	Workshop in Marketing

System Dynamics

15.874	System Dynamics for Business Policy
15.878	Economic Dynamics

Corporate Strategy and Policy

15.931	Strategic Management
15.932	Technology Strategy
15.933	Advanced Strategic Management
15.939	Advanced Topics in Policy and Strategy
15.964	Strategy Models

Appendix C: <u>The Program Used to Compile the Results of the Student Survey.</u>

I wrote a PASCAL program, PROCESSSURVEYS, to compile the data from the returned student surveys. The program takes the raw data and creates a conflict "matrix". This matrix contains, for each pair of courses (i,i'), the number of students who listed both courses on their surveys. The program also keeps track of the total number of students selecting each course. The program uses this data to output a report listing the absolute and relative overlap (defined in the text) for each pair of courses.

The program is implemented in PASCAL on the Sloan School's PR1ME 850 computer. Total run time is less than one minute. The code is extensively documented with comments which include a full data dictionary and description of each module. Section C.1 presents a general discussion of issues relating to the program. Section C.2 discusses some implementation specifics on the PR1ME and Section C.3 presents the actual code. It should be pointed out that this appendix and the program documentation presupposes at least a minimum of programming experience on the part of the reader.

C.1. General Discussion.

The program is extensively documented by comments within the code. However, much of what the code actually does is merely translate course numbers from integer format (e.g., 15xxx) to and from a special PASCAL TYPE defined in the program (e.g., C15xxx). I need to do this because there are several arrays and variables that are defined over all of the courses offered in any given semester. PASCAL does allow the programmer to define a new TYPE, in this case the set of courses offered in any given semester. However, it does not allow that TYPE to consist of non-consecutive integers (e.g., 15001, 15011, 15018, etc.). Thus, I have had to define a new TYPE consisting of (C15001, C15011, C15018, etc.).

The main problem with this method is that like most languages, PASCAL does not have pre-defined input/output procedures for user-defined types. Thus, the program reads the survey data in integer form, converts it to the new TYPE, processes it, and converts it back to integer form for output purposes. This is not very complicated, nor is it difficult to implement. However, it is somewhat inelegant in that I have implemented the conversions with large CASE statments. (CASE statements are a convenient method of implementing a large number of nested IF statements in

PASCAL.)

The current implementation requrires a user to update the following parts of the program each semester. (See the program comments for a more detailed explanation of each of these.)

1. CONSTANTS: FIRST and LAST; FIRSTCOURSE and LASTCOURSE.

FIRST and LAST are global constants in the program. They refer to the integers corresponding to the first and last sequence numbers of the courses offered in any semester. (*e.g.*, 15001 and 15999). FIRSTCOURSE and LASTCOURSE are global variables that are initialized in the MAIN body of the program. (*e.g.*, C15001 and C15999).

2. The COURSES Type.

This is the special user-defined TYPE that contains all of the courses offered in the special format C15xxx. The program currently contains the courses offered for the Fall, 1987 semester. In the future, the user need only change that list.

3. CASE Statements.

Two CASE statements contain the full list of courses. They are found in Procedure READSURVEYS (which converts integer data to C15xxx data) and in Function CHTONUM (Character to number; converts C15xxx to integer form). The user should note that once the COURSE type is updated, an EMACS macro can be easily written to update the other CASE statements all at once. (EMACS is the full-screen editor on the PR1ME; see the PR1ME documentation for details on its use.)

C.2. PR1ME Implementation Specifics.

This section assumes that the reader is somewhat familiar with compiling and executing programs written in third generation languages such as PASCAL and has some experience with PR1MOS, the operating system on the PR1ME. One (but by no means the only) good PASCAL reference is Findlay and Watt (1978). Specific information relating to PASCAL on the PR1ME and other PR1MOS issues can be found in the PR1ME reference manual. (PRIME, 1980).

C.2.a. Compilation.

The source code of the program is currently in a file called CONFLICTS. When that file is updated each semester, it will need to be recompiled. PR1MOS compiles the source code in the CONFLICTS file and creates a file with the compiled (binary) object code called B_CONFLICTS. The PR1MOS command to compile a PASCAL program is "PASCAL FILENAME" where FILENAME is the name of the file containing the source code. (On-line help is available; type "HELP PASCAL" at the "OK" prompt.) The following is the actual compilation command and response indicating no errors. (Here and throughout this Appendix, user commands are in lower case; PR1MOS responses are in UPPER case. Note that PR1MOS actually does <u>not</u> distinguish between upper and lower case characters. Finally, "OK" is the standard PR1MOS "ready" prompt.)

```
OK, pascal conflicts
[PASCAL Rev. 19.4.10]
0000 ERRORS [PASCAL Rev. 19.4.10]
OK,
```

The program should compile successfully as long as no incorrect changes are made. However, since the implementation does require a user to modify the source code, I recommend that the user be experienced at least with some programming language, and preferably PASCAL.

C.2.b. <u>Linking and Loading.</u>

The compiler creates a file called B_FILENAME which contains the binary object code resulting from the source code in FILENAME. (In this case, the actual FILENAME is CONFLICTS.) Once this is accomplished, the user needs to link the object code to other PRIMOS runtime routines. The user does this by using the PRIMOS SEG command. (Again, on-line help can be accessed by typing "HELP SEG".) The following is the appropriate sequence of commands.

```
OK, seg -load
[SEG Rev. 19.4.10]
$ load b_conflicts
$ li paslib
$ li
LOAD COMPLETE
$ save
$ quit
OK,
```

This command sequence creates a SEGMENT sub-directory with the files necessary to run the program. The sub-directory will be called B_FILENAME.SEG.

C.2.c. Execution.

Once the segment directory is created, the user can execute the program by using the following command:

```
OK, seg b_conflicts
69 STUDENT SURVEYS WERE PROCESSED.

OK,
```

If all of the input data is valid, the program will send a message to the terminal indicating the number of surveys that were processed. If some of the input data is invalid (e.g., non-existent course numbers), the program will send appropriate error messages to the terminal.

C.2.d. Input Data.

The program expects the survey data to reside in a text file called SURVEYS. This file can be created in EMACS. It should contain the student survey data, listing the courses for one student per line, in integer form. A maximum of 5 per courses per student are allowed, but fewer than 5 can be entered if the student listed less than 5 courses. Within a line, the data can be "free form", i.e., additional spaces can be placed between the courses. An example of valid input for three students is:

```
15568  15933  15435  15825  15932
15436  15437  15525  15438
15962  15437  15435  15436  15932
```

In this case, the second student listed only four courses. Note that there should not be any blank lines or extraneous characters, comments, etc. in the file.

C.2.e. Program Output.

If all of the input data is valid, the program will compute all of the pairwise conflicts and create a file called GROUPINGS. This file will contain a report which summarizes the conflict data. (Note: every time the program is run, the former GROUPINGS file, if present, will be erased. If the old GROUPINGS file is desired to be kept, it must be renamed.) The GROUPINGS file can either be printed or examined on the screen using EMACS. The following is a sample of the first few lines of the report:

```
                    COURSE CONFLICT REPORT

COURSE 1  ENROLLMENT  COURSE 2  ENROLLMENT  OVERLAP  OVERLAP
                                                       PCT

15013         6        15035         3         1      11.11
15013         6        15435        21         4      14.81
15013         6        15525         8         2      14.29
15013         6        15564         5         3      27.27
15013         6        15568        19         1       4.00
15013         6        15812         3         1      11.11
15013         6        15825        11         1       5.88
15013         6        15931         7         1       7.69
15013         6        15932        24         1       3.33
15013         6        15933        15         3      14.29
15018         5        15231         4         1      11.11
15018         5        15351        15         2      10.00
15018         5        15521         8         1       7.69
15018         5        15615         8         2      15.38
15018         5        15665         3         1      12.50
15018         5        15825        11         1       6.25
15018         5        15832        11         2      12.50
15034         1        15436        16         1       5.88

                        (etc.)
```

The columns represent the following:

- COURSE 1 and COURSE 2 are the two courses in this pair;

- The ENROLLMENT columns list their respective (expected) enrollments; i.e., the number of students listing that course on their surveys;

- OVERLAP is the actual number of students listing both of the courses in the pair ("absolute overlap");

- OVERLAP PCT is the absolute overlap expressed as a percentage of the total number of students listing either or both of the courses ("relative overlap"); (As explained in the text, the maximum relative overlap is 50%).

The output may be made more meaningful by sorting the conflict pairs by their absolute and/or relative overlaps. The sorting can be performed using the PR1MOS SORT command. I will provide the commands below, but again, on-line help is available by typing "HELP SORT".

The SORT command will sort the lines (records) in the report according to any specified columns (fields). In the GROUPINGS file, the absolute overlap field is in columns 57 to 59 and the relative overlap field is in columns 72 to 76. The user types "SORT" at the "OK" prompt. PR1MOS will ask for the input file (GROUPINGS), an output file (say, GROUPINGS.ABS), and the number of fields used for the sort. (e.g., two: absolute and relative overlap.) The default is for an ascending sort, but if the user desires the output to be in descending order (highest conflict pair first), he can specify the 'r' (reverse) option for each set of columns.

The first dialogue shows the command sequence used if the user desires to sort on two pairs of columns: first on absolute overlap, and then on relative overlap for pairs within groups having equal absolute overlap. The sorts will be in reverse order and the output will be sent to a new file called GROUPINGS.ABS. (Note that text files are in ASCII format so the default data type is appropriate.)

```
OK, sort

SORT PROGRAM PARAMETERS ARE:
  INPUT TREE NAME -- OUTPUT TREE NAME FOLLOWED BY
  NUMBER OF PAIRS OF STARTING AND ENDING COLUMNS.

groupings groupings.abs 2

  INPUT PAIRS OF STARTING AND ENDING COLUMNS
  ONE PAIR PER LINE--SEPARATED BY A SPACE.
  FOR REVERSE SORTING ENTER "R" AFTER DESIRED
  ENDING COLUMN--SEPARATED BY A SPACE.
  FOR A SPECIFIC DATA TYPE ENTER THE PROPER CODE
  AT THE END OF THE LINE--SEPARATED BY A SPACE.
     "A"  - ASCII
     "I"  - SINGLE PRECISION INTEGER
     "F"  - SINGLE PRECISION REAL
     "D"  - DOUBLE PRECISION REAL
     "J"  - DOUBLE PRECISION INTEGER
     "U"  - NUMERIC ASCII,UNSIGNED
     "LS" - NUMERIC ASCII,LEADING SEPARATE SIGN
     "TS" - NUMERIC ASCII,TRAILING SEPARATE SIGN
     "LE" - NUMERIC ASCII,LEADING EMBEDDED SIGN
     "TE" - NUMERIC ASCII,TRAILING EMBEDDED SIGN
     "PD" - PACKED DECIMAL
     "AU" - ASCII, UPPER & LOWER CASE SORT EQUAL
     "UI" - UNSIGNED INTEGER
  DEFAULT IS ASCII.

57 59 r
72 76 r

BEGINNING SORT



 PASSES       2           ITEMS      219

[SORT-REV19.2]
OK,
```

The second dialogue sorts first on relative overlap, and then on absolute overlap for pairs within groups having equal relative overlap. In this case, the output is sent to a new file called GROUPINGS.REL.

```
OK, sort

SORT PROGRAM PARAMETERS ARE:
  INPUT TREE NAME -- OUTPUT TREE NAME FOLLOWED BY
  NUMBER OF PAIRS OF STARTING AND ENDING COLUMNS.

groupings groupings.rel 2

  INPUT PAIRS OF STARTING AND ENDING COLUMNS
  ONE PAIR PER LINE--SEPARATED BY A SPACE.
  FOR REVERSE SORTING ENTER "R" AFTER DESIRED
  ENDING COLUMN--SEPARATED BY A SPACE.
  FOR A SPECIFIC DATA TYPE ENTER THE PROPER CODE
  AT THE END OF THE LINE--SEPARATED BY A SPACE.
     "A"  - ASCII
     "I"  - SINGLE PRECISION INTEGER
     "F"  - SINGLE PRECISION REAL
     "D"  - DOUBLE PRECISION REAL
     "J"  - DOUBLE PRECISION INTEGER
     "U"  - NUMERIC ASCII,UNSIGNED
     "LS" - NUMERIC ASCII,LEADING SEPARATE SIGN
     "TS" - NUMERIC ASCII,TRAILING SEPARATE SIGN
     "LE" - NUMERIC ASCII,LEADING EMBEDDED SIGN
     "TE" - NUMERIC ASCII,TRAILING EMBEDDED SIGN
     "PD" - PACKED DECIMAL
     "AU" - ASCII, UPPER & LOWER CASE SORT EQUAL
     "UI" - UNSIGNED INTEGER
  DEFAULT IS ASCII.

72 76 r
57 59 r

BEGINNING SORT



 PASSES       2           ITEMS      219

[SORT-REV19.2]
OK,
```

C.3. Listing of the PASCAL code.

The following pages present a listing of the PASCAL code. It is long, but much of the length is due to extensive comments.

Note: A programmer would probably prefer to work from a version of the program printed directly from the PR1ME on standard computer paper. This output is provided solely for purposes of documentation.

```
PROGRAM PROCESSSURVEYS  (SURVEYS, GROUPINGS, OUTPUT);

(*  This program will read the student survey data
    from the file SURVEYS.

    First, Procedure READSURVEYS will check the course
    numbers that are input to make sure they are valid.
    Valid course numbers are converted to the special
    COURSES type and then read into the array STUDENT.

    Next, the Procedure CALCCONFLICTS computes the number
    of student conflicts for each pair of courses.

    Finally, Procedure FINDPAIRS writes a report summarizing
    all of the absolute and relative conflicts to the
    output file GROUPINGS.  (See thesis text for an
    explanation of the absolute and relative overlap
    measures.)                                          *)


(* --------------------------------------------------- *)


(* The following CONSTANTS must be updated each semester:

   MAXCOURSES is the number of courses each student
   is allowed to list on his/her survey.

   MAXSTUDENTS is an upper bound on the number of
   students returning surveys.  It is used to define
   several arrays so it should not be too large.

   FIRST is the integer representing the lowest numbered
   course offered this semester.

   LAST is the integer representing the highest numbered
   course offered this semester.
                                                          *)

CONST
 MAXCOURSES = 5;
 MAXSTUDENTS = 500;
 FIRST = 15001;
 LAST = 15965;
```

```
(* The user-defined type COURSES contains all of the
   (elective) courses offered this semester.  (CORE
   courses are excluded because the model will not
   attempt to schedule around second year students'
   conflicts resulting from their not finishing the
   CORE in their first years.

   The format is C15xxx where 15xxx is the integer
   course number.  This special type must be used
   since PASCAL will not allow arrays to be indexed
   by integers that are not in consecutive intervals.

   It is true that we could define the relevant arrays for
   every integer value in the interval [15001, 15999].
   However, this would be extremely inefficient since there
   are 1000 integers in this interval, but usually no more
   than 75 electives in any given semester.  (Not every
   number is used.)
                                                        *)


TYPE
 COURSES = ( C15001,
             C15013,
             C15018,
             C15034,
             C15035,
             C15053,
             C15058,
             C15059,
             C15065,
             C15075,
             C15081,
             C15083,
             C15099,
             C15141,
             C15221,
             C15231,
             C15232,
             C15301,
             C15312,
             C15317,
             C15351,
             C15361,
             C15371,
             C15412,
             C15413,
             C15415,
             C15435,
             C15436,
             C15437,
                C15438,
                C15521,
                C15525,
                C15539,
                C15564,
                C15565,
                C15568,
                C15601,
                C15615,
                C15664,
                C15665,
                C15671,
                C15674,
                C15691,
                C15763,
                C15768,
                C15769,
                C15812,
                C15814,
                C15824,
                C15825,
                C15832,
                C15874,
                C15878,
                C15931,
                C15932,
                C15933,
                C15962,
                C15964,
                C15965 );

(* ******************************************** *)
(*                                              *)
(*             DATA DICTIONARY                  *)
(*                                              *)
(* ******************************************** *)

(* The following (global) variables are defined:

   SURVEYS is the text file containing the student survey
   input data.  The data should be in integer format with
   one student per line.  The number of courses can vary
   by student, and within a line, the data can be freeform.

   GROUPINGS is the output text file to which the report
   summarizing the student conflicts is written.
```

```
ENROLL is an array indexed by COURSES.
It holds the number of students expected to enroll in
each course, based on the survey data.

OVERLAP is a two dimensional array, with both dimensions
indexed by COURSES.  For every pair of courses (i,j),
OVERLAP contains the number of students who listed both
i and j on their surveys.  Note that due to the way
OVERLAP is calculated (see procedure FINDPAIRS), the
total overlap between i and j is equal to the sum of

       OVERLAP[i,j] + OVERLAP[j,i].


II and JJ are two index variables defined on the
COURSES type.

NUMSTUDENTS is the actual number of students who
have returned surveys.

STUDENT is the array into which the raw survey data
is read.  For each student, it contains the courses
listed by that student.  (Note that STUDENT contains
the course data in the user-defined COURSE type.)

VALIDCOURSE contains the number of valid courses each
student's survey contained.

ALLVALID is a boolean variable which equals TRUE if
and only if all of the courses read from all of the
student survey data are valid, that is, that they
are in the COURSES type and hence, are actually
being offered this semester.  If ALLVALID is FALSE,
the program will notify the user of the invalid
input data and will skip the overlap calculations.


FIRSTCOURSE and LASTCOURSE are variables defined on
the COURSE type.  They correspond to the lowest
and highest course numbers offered this semester.
FIRSTCOURSE and LASTCOURSE must be updated each
semester in the initialization section of MAIN,
below.
                                                    *)
```

```
VAR
 SURVEYS, GROUPINGS : TEXT;
 ENROLL : ARRAY [COURSES] OF 0..MAXSTUDENTS;
 OVERLAP: ARRAY [COURSES, COURSES] OF 0..MAXSTUDENTS;
 II,JJ : COURSES;
 NUMSTUDENTS : 0..MAXSTUDENTS;
 STUDENT : ARRAY [1..MAXSTUDENTS, 1..MAXCOURSES] OF COURSES;
 VALIDCOURSE : ARRAY [1..MAXSTUDENTS] OF 0..MAXCOURSES;
 ALLVALID : BOOLEAN;
 FIRSTCOURSE, LASTCOURSE : COURSES;


(* -------------------------------------------------- *)
(* -------------------------------------------------- *)


PROCEDURE READSURVEYS;

(* ************************************************** *)

(* This Procedure will process the SURVEYS input file.
   In particular, it will

     1.  Check to make sure that all course input numbers
         are valid.  If all survey input is valid, it will
         return a value of TRUE to ALLVALID.  If not, it
         will return a value of FALSE to ALLVALID and print
         error messages to the OUTPUT file identifying
         specific errors.  (Unless otherwise specified by
         a programmer at a later time, the default OUTPUT
         file is the terminal.)


     2.  Count the number of students responding and
         return that value to the variable NUMSTUDENTS.


     3.  For each student:

           a.  Return the number of valid courses to the
               array VALIDCOURSE.

           b.  Convert the numerical input to the special
               COURSES type and return each student's
               survey response to the array STUDENT.

                                                        *)
 (*****************************************************)
```

```
(* The following local variables are used:

      VALID is a BOOLEAN variable which is TRUE if
      this student's survey is valid in that all of
      courses selected are actually in the COURSES
      type and are thus being offered this semester.
      If VALID is FALSE, that student's data is
      not valid.  In that case, ALLVALID is set to
      FALSE and error messages ar sent to the terminal.


     NEXTCOURSE is a temporary integer variable used to
     store the data read from the SURVEYS file in
     integer form.  A CASE statement converts the integer
     value to the COURSES type to be stored in the
     STUDENT array.


     STUNUM is an integer index variable.  It is used to
     index the processing of the student surveys.
                                                     *)


VAR
 VALID : BOOLEAN;
 NEXTCOURSE : FIRST..LAST;
 STUNUM : 0..MAXSTUDENTS;


BEGIN

 (* First initialize STUNUM and ALLVALID. *)

 STUNUM := 0;
 ALLVALID := TRUE;


 (* Now cycle through the SURVEYS file.  The steps are:

     1.  Initialize VALIDCOURSE for each student and
         increment STUNUM.


     2.  For each student:

           a.  First assume the data is valid.
               (VALID := TRUE)

           b.  Read the next course.

           c.  Process using a CASE statement.  In
```

```
                    particular, if the NEXTCOURSE represents
                    a valid course, convert NEXTCOURSE to
                    the COURSES type and update the STUDENT
                    array for this STUNUM.  If NEXTCOURSE
                    is not a valid course, set VALID to
                    FALSE and write an error message to
                    the terminal.

                 d.  Finally, if VALID is TRUE, update the
                     ENROLL array and the VALIDCOURSE array.

                                                          *)
WHILE NOT EOF(SURVEYS) DO
 BEGIN
   STUNUM := STUNUM + 1;
   VALIDCOURSE[STUNUM] := 0;
   WHILE NOT EOLN(SURVEYS) DO
     BEGIN
       VALID := TRUE;
       READ (SURVEYS, NEXTCOURSE);

       (***********************************************)
       (*                                             *)
       (*  IMPORTANT:  THIS CASE STATEMENT MUST BE    *)
       (*              UPDATED EACH SEMESTER FOR      *)
       (*              THE DIFFERENT SET OF COURSE    *)
       (*              OFFERINGS.                     *)
       (*                                             *)
       (*              SEE THESIS, APPENDIX C.        *)
       (*                                             *)
       (***********************************************)

       CASE NEXTCOURSE OF

15001: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15001;
15013: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15013;
15018: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15018;
15034: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15034;
15035: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15035;
15053: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15053;
15058: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15058;
15059: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15059;
15065: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15065;
15075: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15075;
15081: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15081;
15083: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15083;
15099: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15099;
15141: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15141;
15221: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15221;
15231: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15231;
15232: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15232;
```

```
15301: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15301;
15312: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15312;
15317: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15317;
15351: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15351;
15361: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15361;
15371: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15371;
15412: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15412;
15413: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15413;
15415: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15415;
15435: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15435;
15436: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15436;
15437: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15437;
15438: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15438;
15521: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15521;
15525: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15525;
15539: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15539;
15564: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15564;
15565: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15565;
15568: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15568;
15601: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15601;
15615: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15615;
15664: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15664;
15665: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15665;
15671: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15671;
15674: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15674;
15691: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15691;
15763: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15763;
15768: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15768;
15769: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15769;
15812: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15812;
15814: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15814;
15824: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15824;
15825: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15825;
15832: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15832;
15874: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15874;
15878: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15878;
15931: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15931;
15932: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15932;
15933: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15933;
15962: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15962;
15964: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15964;
15965: STUDENT [STUNUM, (VALIDCOURSE[STUNUM]+1) ] := C15965;

      OTHERWISE
       BEGIN
         VALID := FALSE;
         ALLVALID := FALSE;
         WRITELN ('INPUT FOR STUDENT ', STUNUM:4, ', COURSE ',
                 NEXTCOURSE:7, ' IS INVALID')
       END  (* OTHERWISE *)
      END;  (* OF CASE STATEMENT *)
```

```
        IF VALID THEN
          BEGIN
           VALIDCOURSE[STUNUM] := VALIDCOURSE[STUNUM] + 1;
           ENROLL[STUDENT[STUNUM, VALIDCOURSE[STUNUM]]] :=
            ENROLL[STUDENT[STUNUM, VALIDCOURSE[STUNUM]]] + 1
          END

       END;      (* OF EOLN STATEMENT *)

     READLN (SURVEYS)

   END;          (* OF EOF STATEMENT  *)

   NUMSTUDENTS := STUNUM;

END;            (* OF READSURVEYS    *)


(* --------------------------------------------------- *)
(* --------------------------------------------------- *)

PROCEDURE CALCCONFLICTS;

(* *************************************************** *)
(*
   This procedure processes the STUDENT array to compute,
   for every course pair, the number of students listing
   both of the courses in the pair on their survey.
   Results are placed in the array OVERLAP.

                                                       *)
(* *************************************************** *)


(* The following integer index variables are used:

     I and J are index variables used to cycle through
     the course dimension of the STUDENT array.

     STUNUM is an index variable used to cycle through
     each student in the STUDENT array.
                                                       *)


VAR
 I,J : 1..MAXCOURSES;
 STUNUM : 0..MAXSTUDENTS;
```

```
(* The procedure works as follows:

     1.  We loop through all the students.

     2.  For each student, we loop through all
         of his courses (up to the next-to-last).

     3.  For each course, we loop through all of
         the courses the student listed, after that
         course.  Thus, on each iteration, we have
         identified one overlap.  The OVERLAP array
         is updated accordingly.
                                                  *)

BEGIN
 FOR STUNUM := 1 TO NUMSTUDENTS DO
  FOR I := 1 TO ( VALIDCOURSE[STUNUM] - 1 ) DO
   FOR J := (I+1) TO VALIDCOURSE[STUNUM] DO

    OVERLAP [(STUDENT[STUNUM,I]), (STUDENT[STUNUM,J])] :=
     OVERLAP [(STUDENT[STUNUM,I]), (STUDENT[STUNUM,J])] + 1;


END;  (* OF CALCONFLICTS *)

(* ----------------------------------------------------- *)
(* ----------------------------------------------------- *)

FUNCTION CHTONUM (CHCOURSE : COURSES) : INTEGER;

(* This function converts a variable of the type COURSE to
   its corresponding integer value.  This is necessary
   since PASCAL does not support I/O for user-defined
   types.  Thus, in order to output the conflict pairs,
   the COURSE type must be converted back to integers.   *)


(* TEMPVAL  is used to temporarily store the integer value
   of the input variable, CHCOURSE.  *)

VAR
 TEMPVAL : INTEGER;


(* This CASE statement converts the COURSE  type to the
   appropriate integer value.                         *)
```

```
        (**********************************************)
        (*                                            *)
        (*  IMPORTANT:  THIS CASE STATEMENT MUST BE   *)
        (*              UPDATED EACH SEMESTER FOR     *)
        (*              THE DIFFERENT SET OF COURSE   *)
        (*              OFFERINGS.                    *)
        (*                                            *)
        (*              SEE THESIS, APPENDIX C.       *)
        (*                                            *)
        (**********************************************)


BEGIN
  CASE CHCOURSE OF
    C15001:  TEMPVAL :=  15001;
    C15013:  TEMPVAL :=  15013;
    C15018:  TEMPVAL :=  15018;
    C15034:  TEMPVAL :=  15034;
    C15035:  TEMPVAL :=  15035;
    C15053:  TEMPVAL :=  15053;
    C15058:  TEMPVAL :=  15058;
    C15059:  TEMPVAL :=  15059;
    C15065:  TEMPVAL :=  15065;
    C15075:  TEMPVAL :=  15075;
    C15081:  TEMPVAL :=  15081;
    C15083:  TEMPVAL :=  15083;
    C15099:  TEMPVAL :=  15099;
    C15141:  TEMPVAL :=  15141;
    C15221:  TEMPVAL :=  15221;
    C15231:  TEMPVAL :=  15231;
    C15232:  TEMPVAL :=  15232;
    C15301:  TEMPVAL :=  15301;
    C15312:  TEMPVAL :=  15312;
    C15317:  TEMPVAL :=  15317;
    C15351:  TEMPVAL :=  15351;
    C15361:  TEMPVAL :=  15361;
    C15371:  TEMPVAL :=  15371;
    C15412:  TEMPVAL :=  15412;
    C15413:  TEMPVAL :=  15413;
    C15415:  TEMPVAL :=  15415;
    C15435:  TEMPVAL :=  15435;
    C15436:  TEMPVAL :=  15436;
    C15437:  TEMPVAL :=  15437;
    C15438:  TEMPVAL :=  15438;
    C15521:  TEMPVAL :=  15521;
    C15525:  TEMPVAL :=  15525;
    C15539:  TEMPVAL :=  15539;
    C15564:  TEMPVAL :=  15564;
    C15565:  TEMPVAL :=  15565;
    C15568:  TEMPVAL :=  15568;
    C15601:  TEMPVAL :=  15601;
```

```
     C15615:  TEMPVAL :=  15615;
     C15664:  TEMPVAL :=  15664;
     C15665:  TEMPVAL :=  15665;
     C15671:  TEMPVAL :=  15671;
     C15674:  TEMPVAL :=  15674;
     C15691:  TEMPVAL :=  15691;
     C15763:  TEMPVAL :=  15763;
     C15768:  TEMPVAL :=  15768;
     C15769:  TEMPVAL :=  15769;
     C15812:  TEMPVAL :=  15812;
     C15814:  TEMPVAL :=  15814;
     C15824:  TEMPVAL :=  15824;
     C15825:  TEMPVAL :=  15825;
     C15832:  TEMPVAL :=  15832;
     C15874:  TEMPVAL :=  15874;
     C15878:  TEMPVAL :=  15878;
     C15931:  TEMPVAL :=  15931;
     C15932:  TEMPVAL :=  15932;
     C15933:  TEMPVAL :=  15933;
     C15962:  TEMPVAL :=  15962;
     C15964:  TEMPVAL :=  15964;
     C15965:  TEMPVAL :=  15965

    END;

    CHTONUM := TEMPVAL

 END;

 (* ------------------------------------------------ *)
 (* ------------------------------------------------ *)
```

```
PROCEDURE FINDPAIRS;

(* ************************************************** *)
(*
   This procedure processes the OVERLAP array.
   It outputs a report of all course pair overlaps
   to the GROUPINGS text file.
                                                    *)
(* ************************************************** *)


(* The following local variables are used:

     TOTOVERLAP is the total overlap between two
     courses (i,j).  It equals

         OVERLAP(i,j) + OVERLAP(j,i).


     OVERFACT is the relative overlap percentage for
     each course pair.  (See thesis text for a
     discussion of the relative overlap concept.)


     II and JJ are index variables defined on the
     course type.  They are used to cycle through
     the OVERLAP array.

                                                   *)

VAR
 TOTOVERLAP : INTEGER;
 OVERFACT : REAL;
 II,JJ : COURSES;

(* The procedure merely cycles through the overlap array.
   For each course pair, the absolute and relative overlap
   are computed and written to the output report in the
   GROUPINGS text file.                                   *)


BEGIN

 WRITE(GROUPINGS, '                    ');
 WRITELN(GROUPINGS, 'COURSE CONFLICT REPORT             ');
 WRITELN(GROUPINGS);
 WRITELN(GROUPINGS);
 WRITE(GROUPINGS, 'COURSE 1    ENROLLMENT         ');
 WRITE(GROUPINGS, 'COURSE 2    ENROLLMENT     ');
 WRITELN(GROUPINGS, 'OVERLAP     OVERLAP PCT');
 WRITELN(GROUPINGS);
```

```
FOR II := FIRSTCOURSE TO PRED(LASTCOURSE) DO
  FOR JJ := SUCC(II) TO LASTCOURSE DO

   BEGIN
    TOTOVERLAP := OVERLAP[II,JJ] + OVERLAP[JJ,II];
    IF ((ENROLL[II] + ENROLL[JJ] ) > 0 )
     THEN
      OVERFACT := TOTOVERLAP / (ENROLL[II] + ENROLL[JJ])
      ELSE OVERFACT := 0;
    OVERFACT := OVERFACT * 100;
    IF OVERLAP[II,JJ] > 0 THEN
     BEGIN
       WRITE(GROUPINGS, CHTONUM(II):6, '         ');
       WRITE(GROUPINGS, ENROLL[II]:4, '           ');
       WRITE(GROUPINGS, CHTONUM(JJ):6, '         ');
       WRITE(GROUPINGS, ENROLL[JJ]:4, '      ');
       WRITE(GROUPINGS, TOTOVERLAP:4);
       WRITELN(GROUPINGS, '           ', OVERFACT:6:2)
     END
   END

END;

(* --------------------------------------------------- *)
(* --------------------------------------------------- *)
```

```
BEGIN  (* MAIN *)

 (* INITIALIZATION *)

 (* IMPORTANT:  FIRSTCOURSE AND LASTCOURSE MUST
                BE UPDATED EACH SEMESTER.         *)


 FIRSTCOURSE := C15001;
 LASTCOURSE := C15965;

 FOR II := FIRSTCOURSE TO LASTCOURSE DO
  BEGIN
    ENROLL[II] := 0;
    FOR JJ := FIRSTCOURSE TO LASTCOURSE DO
      OVERLAP[II,JJ] := 0
  END;

 RESET (SURVEYS, 'SURVEYS');
 REWRITE (GROUPINGS, 'GROUPINGS');

 READSURVEYS;
 IF  NOT ALLVALID
  THEN
   WRITELN ('INPUT FOR AT LEAST ONE STUDENT IS INVALID.')
  ELSE
   BEGIN
    WRITE(NUMSTUDENTS:2);
    WRITELN(' student surveys were processed.');
    CALCCONFLICTS;
    WRITELN;
    FINDPAIRS
   END;

 CLOSE(SURVEYS);
 CLOSE(GROUPINGS)
END.
```

Appendix D: Summary of the Model Formulation.

This appendix presents a summary of the model described in the text. All necessary data sets, parameters, and equations are shown. Section D.1 presents the basic model described in Section IV.A. Section D.2 summarizes three alternative formulations: cumulative room size groups, Friday sessions for CORE courses, and specific concentration requirements for different CORE sections. These alternatives are described in the text in Sections IV.B.6 and VII.A.1.

D.1. The Basic Model.

Let x_{ij} = { 1 if course i is assigned to time j;

{ 0 otherwise.

We define the following subsets of all of the courses:

I_r = set of courses compatible with room type r;

I_f = set of courses taught by faculty member f;

I_s = set of courses taken by CORE section s;

I_s^h = set of courses taken by section s in half semester h;

I_c = set of courses fulfilling requirements for concentration c;

I_p = set (pair) of courses in high conflict pair p;

I_{sem} = set of courses offered as three hour seminars;

Now define the eight standard time slots as described in the text (e.g., j = 1 is Mon/Wed at 9; j = 2 is Mon/Wed at 10:30, etc.). Further, define subsets of the standard time slots as follows:

TSEM = set of time slots not valid for three hour seminar courses;

JMW = Monday/Wednesday time slots;

JTT = Tuesday/Thursday time slots;

Define the following subsets of the faculty:

FMW = set of faculty members wishing to teach on Monday/Wednesday;

FTT = set of faculty members wishing to teach on Tuesday/Thursday;

FMWNBB = set of faculty members teaching on Monday/Wednesday who <u>do</u> <u>not</u> wish to teach back-to-back;

FTTNBB = set of faculty members teaching on Tuesday/Thursday who <u>do</u> <u>not</u> wish to teach back-to-back;

FMWYBB = set of faculty members teaching on Monday/Wednesday who <u>do</u> wish to teach back-to-back;

FTTYBB = set of faculty members teaching on Tuesday/Thursday who <u>do</u> wish to teach back-to-back;

Finally, define the following parameters:

c_{ij} = a measure of the desirability of assigning course i to time j (faculty "utility");

$N\{r\}_j$ = the number of rooms of type r available at time j;

The model is thus:

$$\max \quad \sum_i \sum_j c_{ij} x_{ij} \qquad \text{(D1)}$$

subject to:

$$\sum_j x_{ij} = 1 \qquad \text{all } i; \qquad \text{(D2)}$$

$$\sum_{i \in I_r} x_{ij} <= N\{r\}_j \qquad \text{all } r,j \qquad \text{(D3)}$$

$$\sum_{i \in I_s^h} x_{ij} <= 1 \qquad \text{all } s,h,j \qquad \text{(D4)}$$

$$\sum_{i \in I_f} x_{ij} <= 1 \qquad \text{all } f,j \qquad \text{(D5)}$$

$$\sum_{i \in I_c} x_{ij} <= 1 \qquad \text{all } c,j \qquad \text{(D6)}$$

$$\sum_{i \in I_p} x_{ij} <= 1 \qquad \text{all } p,j \qquad \text{(D7)}$$

$$\sum_{i \in I_{sem}} \sum_{j \in TSEM} x_{ij} = 0 \qquad \text{(D8)}$$

$$\sum_{i \in I_f} x_{ij} = 0 \qquad f \in FMW; \quad j \in JTT; \qquad \text{(D9)}$$

$$\sum_{i \in I_f} x_{ij} = 0 \qquad f \in FTT; \quad j \in JMW; \qquad \text{(D10)}$$

$$\sum_{i \in I_f} (x_{i1} + x_{i2}) <= 1 \qquad f \in FMWNBB; \qquad \text{(D11)}$$

$$\sum_{i \in I_f} (x_{i3} + x_{i4}) <= 1 \qquad f \in FMWNBB; \qquad \text{(D12)}$$

$$\sum_{i \in I_f} (x_{i5} + x_{i6}) <= 1 \qquad f \in FTTNBB; \qquad \text{(D13)}$$

$$\sum_{i \in I_f} (x_{i7} + x_{i8}) <= 1 \qquad f \in FTTNBB; \qquad \text{(D14)}$$

$$\sum_{i \in I_f} (x_{i1} + x_{i3}) <= 1 \qquad f \in \text{FMWYBB}; \qquad \text{(D15)}$$

$$\sum_{i \in I_f} (x_{i1} + x_{i4}) <= 1 \qquad f \in \text{FMWYBB}; \qquad \text{(D16)}$$

$$\sum_{i \in I_f} (x_{i2} + x_{i3}) <= 1 \qquad f \in \text{FMWYBB}; \qquad \text{(D17)}$$

$$\sum_{i \in I_f} (x_{i2} + x_{i4}) <= 1 \qquad f \in \text{FMWYBB}; \qquad \text{(D18)}$$

$$\sum_{i \in I_f} (x_{i5} + x_{i7}) <= 1 \qquad f \in \text{FTTYBB}; \qquad \text{(D19)}$$

$$\sum_{i \in I_f} (x_{i5} + x_{i8}) <= 1 \qquad f \in \text{FTTYBB}; \qquad \text{(D20)}$$

$$\sum_{i \in I_f} (x_{i6} + x_{i7}) <= 1 \qquad f \in \text{FTTYBB}; \qquad \text{(D21)}$$

$$\sum_{i \in I_f} (x_{i6} + x_{i8}) <= 1 \qquad f \in \text{FTTYBB}; \qquad \text{(D22)}$$

x binary;

The equations represent the following conditions or policies. (A detailed discussion is given in Sections IV.A.1 and IV.A.2).

(D1). Objective Function.

(D2). Assignment of every course to one slot.

(D3). Room availability.

(D4). CORE section feasibility.

(D5). Faculty feasibility.

(D6). No more than one course per concentration at any one time.

(D7). High overlap courses not scheduled concurrently.

(D8). Three hour seminars given at 2:30 PM.

(D9). Faculty members teaching only on Monday/Wednesday.

(D10). Faculty members teaching only on Tuesday/Thursday.

(D11), (D12). Faculty members teaching on Mon/Wed who <u>do</u> <u>not</u> wish to teach back-to-back.

(D13), (D14). Faculty members teaching on Tue/Thur who <u>do</u> <u>not</u> wish to teach back-to-back.

(D15), (D16), (D17), and (D18).

Faculty members teaching on Monday/Wednesday who <u>do</u> wish to teach back-to-back.

(D19), (D20), (D21), and (D22).

Faculty members teaching on Tuesday/Thursday who <u>do</u> wish to teach back-to-back.

D.2. Alternative Formulations.

This section considers three alternative formulations: cumulative room size groupings; Friday sessions for certain CORE courses; and CORE requirements that vary by concentration.

D.2.a. Cumulative Room Size Groups.

This alternative was discussed in Section IV.B.6 in the text. The formulation would allow smaller courses to be scheduled in larger rooms, if such rooms were available. To implement this formulation, we would change the room availability constraint (D3) as follows:

$$\sum_{i \in I_{R1}} x_{ij} <= N\{1\}_j \qquad \text{all } j; \qquad \text{(D3a)}$$

$$\sum_{\substack{i \in I_{R1} \\ i \in I_{R2}}} x_{ij} <= (N\{1\}_j + N\{2\}_j) \qquad \text{all } j; \qquad \text{(D3b)}$$

$$\sum_{\substack{i \in I_{R1} \\ i \in I_{R2} \\ i \in I_{R3}}} x_{ij} <= (N\{1\}_j + N\{2\}_j + N\{3\}_j) \qquad \text{all } j; \qquad \text{(D3c)}$$

$$\sum_{\substack{i \in I_{R1} \\ i \in I_{R2} \\ i \in I_{R3} \\ i \in I_{R4}}} x_{ij} <= (N\{1\}_j + N\{2\}_j + N\{3\}_j + N\{4\}_j) \qquad \text{all } j; \qquad \text{(D3d)}$$

In this revised formulation, constraint (D3a) ensures that enough rooms of R1 are available. Constraint (D3b) allows courses in group R2 to be scheduled in rooms in either R2 or R1. Rooms in R1 can be used only if there is slack in constraint (D3a). Similarly, constraint (D3c) allows courses in R3 to be scheduled in R3, R2, or R1, again if there is slack in constraints (D3a) and (D3b). Finally, constraint (D3d) allows courses in R4 to be scheduled in any available slot, again subject to availability in R1, R2, R3, as measured by the slack in the other constraints.

D.2.b. Friday Sessions for CORE Courses.

This alternative was discussed in the text in Section VII.A.1. Let I_{sf}^{h} be the set of courses required for CORE section s in half semester h that have Friday sessions. As described in the text, the policy is to ensure that the Friday sessions are held at the same time as the sessions during the rest of the week. Thus, let

J_1 = { Mon/Wed 9-10:30 and Tue/Thur 9-10:30 }

J_2 = { Mon/Wed 10:30-12 and Tue/Thur 10:30-12 }

J_3 = { Mon/Wed 1-2:30 and Tue/Thur 1-2:30 }

J_4 = { Mon/Wed 2:30-4 and Tue/Thur 2:30-4 }.

Then the constraints are:

$$\sum_{i \in I^h_{st}} \sum_{j \in J_t} x_{ij} <= 1 \qquad \text{all s, h, t;} \qquad \text{(D23)}$$

D.2.c. <u>CORE Requirements Vary by Concentration.</u>

As discussed in the text in Section VII.A.1, this condition does not require any new constraints. The requirement that all potential section/concentration combinations be kept feasible can be modelled by simply increasing the number of <u>de facto</u> sections. In particular, if there are N concentrations with special requirements, and if there are S sections, then there will be a total of N * S <u>de facto</u> combinations.

In the text, I recommended that this problem instead be handled by reassigning sections by concentration for the Spring term. If this is done, the number of sections need not change. However, the sets I_s containing the courses required for each section s will change, depending on the concentrations assigned to that section.

Appendix E: Assignment of Sections to Offerings.*

The Sloan School currently divides the first year class into 12 Sections (A thru L). CORE courses are offered in four different size formats which allocate the 12 sections as follows:

Type I:	2 offerings,	6 sections per offering;
Type II:	3 offerings,	4 sections per offering;
Type III:	4 offerings,	3 sections per offering;
Type IV:	6 offerings,	2 sections per offering.

Note the important distinction between a "section" and an "offering". A section is a group of students. An offering is a group of sections assigned to take a given CORE course together. (*E.g.*, Accounting, Sections A-D is an offering.) Professors are assigned to offerings. The choice of a specific size format for each CORE course is a policy decision made well in advance of the actual scheduling process. However, as described in the text, (Section IV.B.2) the model does not assign specific sections or professors to course offerings; such assignments are taken as given by the model.

---

* I would like to acknowledge the assistance provided by Professor Robert Freund in helping me to better understand the issues discussed in this appendix.

Currently, the school assigns sections to offerings in lexicographic order. That is, the sections are assigned to offerings as follows:

Type I:	2 offerings:	A-F and G-L;
Type II:	3 offerings:	A-D, E-H, and I-L;
Type III:	4 offerings:	A-C, D-F, G-I, and J-L;
Type IV:	6 offerings:	A-B, C-D, E-F, G-H, I-J, K-L.

This appendix will show that this rule of thumb is in fact optimal in terms of minimizing the number of what I call "section conflicts". I define "section conflicts" to be the number of offerings, across courses, that include the same sections. The fewer the section conflicts, the more independently can the various offerings of different CORE courses be scheduled. The fact that the rule of thumb is optimal is particularly important given that the model does not endogenously assign sections to courses.

We begin with the Type I courses, those given in two offerings. For convenience, let us assume that sections A-F are assigned to the first offering, and sections G-L to the second. The actual letters are arbitrary at this point. If necessary, the sections could be reordered to achieve this assignment.

Now consider the Type III courses, those given in four offerings. Clearly, an optimal split of A-F and G-L would be one such that each of the four Type III offerings contains sections which belong to only one of the two Type I offerings. In that case, each Type III offering would intersect only one Type I offering. Conversely, each Type I offering would intersect two Type III offerings. Since the number of sections per Type I offering (6) is greater than the number per Type III offering (3), this is the minimum possible.

Therefore, let us assume that the four offerings are allocated as follows:

Type I offering:	A-F;	Type III offerings A-C, D-F;
Type I offering:	G-L;	Type III offerings G-I, J-L;

Again, the assignment of specific sections <u>within</u> a Type I offering to a Type III offering is arbitrary at this point. For example, an allocation of A-F to {A,B,F} and {C,D,E} is equivalent to the one shown. For convenience, let us again reorder the sections (if necessary) to obtain the groupings as shown.

Now consider the Type II courses, those given in three offerings of four sections each. Since 4, unlike 3, is not a divisor of 6, at least one of the Type II offerings will have to overlap two of the Type I offerings. In particular, four of the six sections of each Type I offering can be allocated to one Type II offering. However, at least two (= 6-4) of the six sections in each Type I group must overlap a second Type II offering.

Similarly, consider Type II relative to Type III. Since the number of sections in each Type II offering (4) is greater than the number of sections in each Type III offering (3), each Type II offering must overlap at least two Type III offerings.

Given this analysis, (one of) the way(s) to allocate the the sections for the Type II courses optimally is into offerings that contain the following groups of sections: A-D; E-H; and I-L. The A-D and I-L offerings overlap only one Type I offering. The E-H offering overlaps two Type I groups (E-F belong to A-F and G-H to G-L) but this is the minimum possible, as discussed above. In addition, each Type II offering overlaps two Type III offerings. Again, as discussed above, this is the minimum possible.

Finally, consider the Type IV courses. These courses are given in six offerings of two sections each. Clearly, since 2 divides 6 and 4, the sections can be assigned to

Type IV offerings so as to overlap only one Type I and Type II offering respectively. However, 2 does not divide 3, the number of courses per Type III offering. Thus, at least some of the Type IV offerings will have to overlap more than one Type III offering. Now, while 2 does not divide 3, it does divide (2*3). Thus, the six sections in each of the two groups of two Type III offerings (_i.e._, A-C, D-F; and G-I, J-L) can be assigned so that 3 Type IV offerings account for each of those two Type III offerings. Moreover, since 2 (the number of sections per Type IV offering) is less than 3 (the number of sections per Type III offering), two of those three Type IV offerings need overlap only one Type III offering. Thus, an optimal configuration is:

Type IV Offering	Number of Overlaps with Offerings in:		
	Type I	_Type II_	_Type III_
A-B	1	1	1
C-D	1	1	2
E-F	1	1	1
G-H	1	1	1
I-J	1	1	2
K-L	1	1	1

Thus, the straightforward lexicographic ordering employed by the Sloan School is an optimal way of allocating the sections to offerings. Other combinations could be found, but they can be shown to be combinatorially equivalent to this one, merely by reordering the sections. (_E.g._, switch sections B and C everywhere. The resulting combination is also optimal, but obviously equivalent to the one shown. Note that not every such exchange would maintain optimality. For example, the configuration resulting from an exchange of sections A and L would _not_ be equivalent, nor would it be optimal.)

It is important to emphasize what this analysis _does_ and _does not_ imply. The only conclusion we can draw is that the current method of allocating 12 sections to these 4 different types of courses is optimal in terms of minimizing the number of section overlaps across courses. From a scheduling standpoint, this method is preferred to one in which the sections are assigned to offerings at random. We may conclude that if the administration desired to mix students of different sections in the second semester, it would be preferable, from a scheduling perspective, simply to reassign the students to sections, as opposed to reassigning the sections to offerings.

However, the analysis presented here does _not_ imply that the current (arbitrary) way of allocating offerings (groups of sections) to professors is similarly optimal. In the text, (Section IV.B.2) I discuss that issue/problem in greater detail. In particular, my discussion there acknowledges the conceptual problem, but describes how it may tend to be less of an empirical issue since many professors teaching CORE courses teach more than one offering anyway.

Appendix F: <u>GAMS Model and Documentation.</u>

This appendix presents the actual GAMS model used for the Fall, 1987 test case. In the future, the GAMS code can be updated to obtain schedules for other semesters. Section F.1 <u>briefly</u> describes some essential GAMS syntax and implementation issues. Section F.2 summarizes the data used in the model. Section F.3 describes the equations and relates them to the model summary in Appendix D. Finally, Section F.4 presents the actual GAMS code.

F.1. <u>GAMS Essentials.</u>

It is not my purpose here to provide a comprehensive user's manual for the GAMS language. A reader unfamiliar with GAMS is urged to consult the GAMS manual. (Kendrick and Meeraus, 1985). It should be noted that GAMS was especially designed to help a non-technical user formulate and solve mathematical programs quickly and easily. However, there are a few essentials I can describe quickly so as to help such a reader understand the code for this model.

There are five key components to a GAMS formulation. These are: Sets, Data, Variables, Equations, and Model & Solution Statments. I will briefly discuss each of these.

Sets. GAMS sets correspond to the data sets I have used in the text and Appendix D. For example, I have defined sets of all courses (I), all time slots (J), faculty members (F), and so on.

The reader should note two important facts about sets. First, if a subsequent set is a subset of a former set, that fact should be denoted in the GAMS formulation so that GAMS can perform "domain checking" to ensure that the subset is indeed valid. For example, if I is the set of all courses, and CORE is a subset consisting of the CORE courses, then the definition of CORE should be CORE(I) to denote the fact that CORE is a subset of I.

Second, sets can denote mappings between other sets. For example, I have used a set FTOI(F,I) which maps courses (I) to faculty members (F). (These sets correspond to the sets I_f in the text and Appendix D.) Other mappings I have used include CONTOI(CON,I) (concentrations to courses) and PTOI(P,I) (high conflict course pairs to courses). These correspond, respectively, to I_c and I_p in Appendix D.

Data. Data in GAMS can be scalar, or one or multi-dimensional. One dimensional data (e.g., estimated enrollment by course) is represented by PARAMETERS. Data that is two or more dimensional (e.g., the number of rooms of type r available at time j or the objective contribution of assigning course i to time j) is represented by TABLES. The dimensions of a variable a specified in parentheses when that variable is defined (e.g., ENROLL(I) or CONTRIB(I,J)).

Variables. Variables can also be multi-dimensional. In my model, there are two variables: X(I,J) corresponds to the definition in the text; OBJ is the objective function. (GAMS requires the objective function to be a single variable. In reality, the OBJ variable is only used for "accounting" purposes.)

Equations. Like the variables, the equations in a GAMS model can also be multi-dimensional. This is one of the chief advantages of using GAMS. For example, ASGT(I) is the equation which ensures that all courses (I) are assigned to one time slot. That equation need only be specified once; GAMS will automatically generated the relevant equations for all the courses in the Set I. The equations I defined in the GAMS model correspond directly to the equations in Appendix D.

Model and Solution Statements.

The Model statement tells GAMS which equations to include in the model. In my model, all equations are included. However, sensitivities could be run which would not include certain constraints. In those cases, the model statement would include only those constraints desired for that run. Multiple models can be defined within a given formulation. The SOLVE statement tells GAMS which model(s) to solve on any given run, which solution package to use (e.g., LP), the direction (max or min) of the optimization, and the objective function.

Other Aspects.

Output. A default model output will be generated. The user can tailor output reports using DISPLAY statements.

Comments. Comments in a GAMS formulation are denoted by an asterisk (*) in the first column of a row. All other rows must begin in the second (or beyond) column.

Syntax. Given the explanation above, the model syntax should be straightforward, with one exception. GAMS uses the $ sign to mean "such that". For example, consider the following equation:

```
Faculty(F,J)..     SUM ( I $ FTOI(F,I), X(I,J)) =L= 1;
```

This equation says that, "For each faculty member F and time J, the sum of all the variables X(I,J) corresponding to courses I _such_ _that_ I is in FTOI(F,I) (for this F) must be less than or equal to one." This is the faculty feasibility constraint.

Help. In addition to the user's manual, a file GAMSHELP.HELP is on the PR1ME. That file explains how one may execute a GAMS formulation as well as various GAMS runtime options that are available on the PR1ME.

Once again, a potential user is referred to the GAMS manual (Kendrick and Meeraus, 1985) for further details.

F.2. _Data Used in the Sloan School Model._

The following data sets and parameters are defined in the GAMS model. I give a brief description of each. Unless otherwise noted, the GAMS data corresponds directly to the model in Appendix D.

Data Sets.

1. I is the set of courses.

2. J is the set of time slots.

3. TSEM(J) is the subset of time slots not valid for three hour seminars.

4. JMW(J) is the subset of time slots occurring on Monday/Wednesday.

5. JTT(J) is the subset of time slots occurring on Tuesday/Thursday.

6. R is the set of room types.

7. S is the set of CORE sections.

8. HALF is the set of half semesters. (Corresponds to (h) in Appendix D.)

9. CORE(I) is the subset of courses that are in the CORE. (Not explicitly used in Appendix D; A discussion of the CORE section constraints in GAMS is given below).

10. F is the set of faculty members teaching more than one course during the standard hours. (Faculty members teaching only one course will always be feasible.)

11. FTOI(F,I) is the subset of courses I taught by faculty member F. (Corresponds to I_f in Appendix D.)

12. FMW(F) is the subset of faculty members desiring to teach on Monday/Wednesday.

13. FMWNBB(FMW) is the subset of faculty members teaching on Mon/Wed who do not wish to teach back-to-back.

14. FMWYBB(FMW) is the subset of faculty members teaching on Mon/Wed who do wish to teach back-to-back.

15. FTT(F) is the subset of faculty members desiring to teach on Tuesday/Thursday.

16. FTTNBB(FTT) is the subset of faculty members teaching on Tue/Thur who <u>do</u> <u>not</u> wish to teach back-to-back.

17. FTTYBB(FTT) is the subset of faculty members teaching on Tue/Thur who <u>do</u> wish to teach back-to-back.

18. CON is the set of defined concentrations. (Corresponds to c in Appendix D.)

19. CONTOI(CON,I) is the mapping of courses I to concentration CON. (Corresponds to I_f in Appendix D.)

20. P is the set of pairs of courses that have a high overlap in student demand. (Absolute and Relative).

21. PTOI(P,I) is the mapping of courses I to the high conflict pairs P. (Corresponds to I_p in Appendix D.)

22. SEM(I) is the subset of courses desired to be offered as three hour seminars.

<u>Parameters.</u>

1. CONTRIB(I,J) is the contribution to total faculty satisfaction obtained when course I is assigned to time J. It corresponds to c_{ij} in Appendix D.

2. COREHALF(CORE,HALF) is a parameter which equals 1 if a given CORE course is offered in this HALF semester; it equals zero otherwise. It is used in the CORE section feasibility constraints.

3. COREMAP(CORE,S) is a parameter which equals 1 if a given CORE course is required for Section S; it equals zero otherwise.

The reader may note that the product of COREHALF and COREMAP is a parameter which equals 1 if the course is required for Section S in half-semester HALF. The parameter thus corresponds to I_s^h in Appendix D.

4. ROOMS(J,R) is the number of rooms of type R available at time J. It corresponds to $N\{r\}_j$ in Appendix D.

5. ENROLL(I) is the estimate of expected enrollment for all courses I. It is used to allocate the courses to room types.

6. ROOMMAP(I,R) is a parameter which equals 1 if course I is assigned to room group R; it equals zero otherwise. ROOMMAP is calculated using the ENROLL parameter. It corresponds to I_r in Appendix D.

F.3. Equations in the GAMS Model.

The equations in the GAMS model correspond directly to those presented in Appendix D. In this section, I present, for each equation in Appendix D, the corresponding GAMS equation.

Appendix D Equation	GAMS Equation
D1	OBJDEF
D2	ASGT(I)
D3	ROOMCAP(J,R)
D4	REQCORE(S,HALF,J)
D5	FACULTY(F,J)

Appendix D Equation	GAMS Equation
D6	CONCEN(CON,J)
D7	CONPAIRS(P,J)
D8	SEMINAR
D9	OFFMW(FMW,JTT)
D10	OFFTT(FTT,JMW)
D11	NBBMWAM(FMWNBB)
D12	NBBMWPM(FMWNBB)
D13	NBBTTAM(FTTNBB)
D14	NBBTTPM(FTTNBB)
D15	YBBMW1(FMWYBB)
D16	YBBMW2(FMWYBB)
D17	YBBMW3(FMWYBB)
D18	YBBMW4(FMWYBB)
D19	YBBTT1(FTTYBB)
D20	YBBTT2(FTTYBB)
D21	YBBTT3(FTTYBB)
D22	YBBTT4(FTTYBB)

F.4. The GAMS Code.

The remainder of this appendix presents the actual GAMS code. This code is also found in the file F87NEWV2.GAMS on the PR1ME 850.

```
OPTION ITERLIM = 3000;
OPTION RESLIM = 2000;

SET I   Set of all Courses taught in standard times in Fall 1986

 /

   15011AF    Applied Micro Sections A thru F
   15011GL    Applied Micro Sections G thru L
   15061AF    DSS II Sections A thru F
   15061GL    DSS II Sections G thru L
   15280AB    Communication Sections A and B
   15280CD    Communication Sections C and D
   15280EF    Communication Sections E and F
   15280GH    Communication Sections G and H
   15280IJ    Communication Sections I and J
   15280KL    Communication Sections K and L
   15311AC    Managerial Behavior Sections A thru C
   15311DF    Managerial Behavior Sections D thru F
   15311GI    Managerial Behavior Sections G thru I
   15311JL    Managerial Behavior Sections J thru L
   15515AD    Accounting Sections A thru D
   15515EH    Accounting Sections E thru H
   15515IL    Accounting Sections I thru L
   15560AF    DSS I Sections A thru F
   15560GL    DSS I Sections G thru L
   15930AC    Strategy Sections A thru C
   15930DF    Strategy Sections D thru F
   15930GI    Strategy Sections G thru I
   15930JL    Strategy Sections J thru L
   15001      Managerial Economics 1 and 2
   15013      Industrial Econ for Strategic Decisions
   15018      Econ of Intl Business
   15034      Appl Econometrics and Forecasting for Mgt
   15035      Pricing Strategy
   15053      Introduction to Mgt Science
   15058      Applied Math Programming
   15059      Math Prog Models and Applications
   15065      Decision Analysis
   15075      Applied Statistics
   15081      Introduction to Math Programming
   15083      Combinatorial Optimization
   15099      Doctoral Statistics
   15141      Comparative Health Systems
   15221      Intl Business Mgt
   15231      Mgt and Tech in the People's Republic of China
   15232      The Firm and Business Environment in JapanJ
   15301      Managerial Psychology
   15312      Managerial Decision Making and Leadership
   15317      Comparative Study of Organizations
   15351      Managing Technology and Innovation
   15361      MOT course in MTI
   15371      The R&D Process
   15412      Financial Management II
   15413      Topics in Corporate Financial Management
   15415A     Finance Theory Section A
   15415B     Finance Theory Section B
   15435A     Corporate Financing Decisions Section A
   15435B     Corporate Financing Decisions Section B
   15436      International Managerial Finance
   15437      Options and Futures Markets
   15438      Investment Banking and Markets
   15501A     Financial and Cost Accounting Section A
   15501B     Financial and Cost Accounting Section B
```

```
     15521      Management Accounting and Control
     15525      Corporate Financial Accounting
     15539      Special Seminar in Accounting
     15564      Management Info Technology I
     15565      Management Info Technology II
     15568      Management Info Systems
     15601      The American Legal System 601 and 611
     15615      Mgrs Legal Function Birth to Bankruptcy
     15664      Management of Human Resources (paired with 15665)
*
*
*         15665 was paired with 15664 for scheduling purposes only
*         since both are three hour seminar courses.
*
*
*    15665      Power and Negotiation (not modelled -- included with 15664)
     15671      Labor Economics I
     15674      Comparative Systems of IR and HRM
     15691      Research Seminar in IR
     15763      The Practice of Operations Management
     15768      Operations Mgt in Services Industry
     15769      Manufacturing Strategy
     15812      Marketing Management
     15814      MOT course in Marketing
     15824      Marketing Communications
     15825      Marketing Models
     15832      Marketing Measurement
     15874      System Dynamics for Business Policy
     15878      Economic Dynamics
     15932      Technology Strategy
     15933      Advanced Strategic Management
     15951B     Introductory Managerial Accounting
     15951C     Undergraduate Managerial Communication
     15962      Competition in Telecommunications Industries
     15964      Strategy Models
     15965      Spec Sem in Mgt Technology Mktg Interface

                                                              /


    J  Standard Time Slots

   /
     t1          Monday and Wednesday 9 to 10:30
     t2          Monday and Wednesday 10:30 to 12
     t3          Monday and Wednesday 1 to 2:30
     t4          Monday and Wednesday 2:30 to 4
     t5          Tuesday and Thursday 9 to 10:30
     t6          Tuesday and Thursday 10:30 to 12
     t7          Tuesday and Thursday 1 to 2:30
     t8          Tuesday and Thursday 2:30 to 4    /


    TSEM(J)  Time Slots that are not applicable to three hour seminars

    / t1, t2, t3, t5, t6, t7 /


   JMW(J) Time Slots on Mondays and Wednesdays

   / t1, t2, t3, t4 /
```

```
JTT(J) Time Slots on Tuesdays and Thursdays

/ t5, t6, t7, t8 /


R  Room Types

 /
  R1          Capacity GE 90
  R2          Capacity GE 55 and LE 89
  R3          Capacity GE 25 and LE 54
  R4          Capacity LE 24
     /


S  CORE Sections

/ SecA, SecB, SecC, SecD, SecE, SecF, SecG, SecH, SecI, SecJ, SecK, SecL /


HALF  Half Semesters

/ H1, H2 /


CORE(I)  Courses in the CORE

/
  15011AF    Applied Micro Sections A thru F
  15011GL    Applied Micro Sections G thru L
  15061AF    DSS II Sections A thru F
  15061GL    DSS II Sections G thru L
  15280AB    Communication Sections A and B
  15280CD    Communication Sections C and D
  15280EF    Communication Sections E and F
  15280GH    Communication Sections G and H
  15280IJ    Communication Sections I and J
  15280KL    Communication Sections K and L
  15311AC    Managerial Behavior Sections A thru C
  15311DF    Managerial Behavior Sections D thru F
  15311GI    Managerial Behavior Sections G thru I
  15311JL    Managerial Behavior Sections J thru L
  15515AD    Accounting Sections A thru D
  15515EH    Accounting Sections E thru H
  15515IL    Accounting Sections I thru L
  15560AF    DSS I Sections A thru F
  15560GL    DSS I Sections G thru L
  15930AC    Strategy Sections A thru C
  15930DF    Strategy Sections D thru F
  15930GI    Strategy Sections G thru I
  15930JL    Strategy Sections J thru L   /
```

```
F  Faculty members teaching more than one course

  /
   Berndt
   Barnett
   Yates
   Piotrowski
   Nickel
   Healy
   OBrien
   Treacy
   Malone
   Venkatrama
   Freund
   Kaufman
   Westney
   VonHippel
   Parsons
   Ancona
   Lessard
   Allen
   Bhushan
   Piore
   Kardes
   Horwitch
             /


FTOI(F,I)  Mapping of courses to Faculty teaching more than one course

  /
   Berndt.(15011AF, 15011GL, 15034)
   Barnett.(15061AF, 15061GL)
   Yates.(15280AB, 15280EF)
   Piotrowski.(15280CD, 15280GH)
   Nickel.(15280IJ, 15280KL)
   Healy.(15515EH, 15539)
   OBrien.(15515AD, 15515IL, 15525)
   Treacy.(15560AF, 15560GL)
   Malone.(15560AF, 15560GL)
   Venkatrama.(15930DF, 15930GI)
   Freund.(15059, 15081)
   Kaufman.(15065, 15099)
   Westney.(15232, 15317)
   VonHippel.(15351, 15965)
   Parsons.(15435A, 15435B)
   Ancona.(15311DF, 15361)
   Lessard.(15221, 15436)
   Allen.(15301, 15371)
   Bhushan.(15501A, 15501B, 15951B)
   Piore.(15671, 15674)
   Kardes.(15812, 15832)
   Horwitch.(15932, 15933)
                         /


FMW(F)  Faculty that only want to teach on Mondays and Wednesdays

  / Treacy, Malone, Venkatrama, Freund, Parsons, Kaufman, Ancona,
    Barnett, VonHippel, Lessard, Allen /
```

```
FMWNBB(FMW)  Faculty that do not want to teach back to back on MW

  / Freund, Kaufman /


FMWYBB(FMW) Faculty that do want to teach back to back on MW

  / Treacy, Malone, Parsons, Allen /


FTT(F)  Faculty that only want to teach on Tuesdays and Thursdays

  / Berndt, Healy, OBrien, Westney, Kardes, Horwitch /


FTTNBB(FTT)  Faculty that do not want to teach back to back on TTh

 / Healy, Westney, Kardes /


* FTTYBB(FTT) Faculty that do want to teach back to back on TTh

* This set is "commented out" since it is null this semester.


CON  Defined Concentrations

  /   Acctg      Accounting and Control
      ApplEcon   Applied Economics
      Strategy   Corporate Strategy
      OpRes      Operations Research
      OpMgt      Operations Management
      Finance    Finance
      HRM        Human Resource Management
      OrgStud    Organization Studies
      Mktg       Marketing
      MIS        Management Info Systems
      IR         Industrial Relations
      SysDy      System Dynamics
      Intl       International Business
      Health     Health Care Management
      MTI        Mgt of Tech Innovation
      Law        Business Law
                                         /
```

```
CONTOI (CON,I)  Mapping of Courses to Concentrations

   /
    Acctg.(15521, 15525, 15539)
    ApplEcon.(15013, 15018, 15034, 15035, 15962)
    Strategy.(15932, 15933, 15964)
    Finance.(15435A, 15436, 15437, 15438)
    Health.(15141)
    HRM.(15312, 15664, 15671, 15674)
    IR.(15664, 15671, 15674, 15691)
    Intl.(15018, 15221, 15231, 15232, 15317, 15436, 15674)
    MIS.(15564, 15565, 15568)
    Mktg.(15034, 15035, 15824, 15825, 15832, 15965)
    OpRes.(15059, 15065, 15081, 15083, 15099)
    OpMgt.(15763, 15768, 15769)
    OrgStud.(15312, 15317)
    SysDy.(15874, 15878)
    MTI.(15351, 15371, 15965, 15932)      /



P   Number of pairs of courses that should not be taught together

   / ABS1, ABS2, ABS3, ABS4, ABS5, ABS6, ABS7, ABS8, ABS9,
     REL1, REL2, REL3, REL4, REL5, REL6, REL7, REL8  /



PTOI(P,I)  Mapping of course pairs to courses

  / ABS1.(15568, 15932)
    ABS2.(15351, 15568)
    ABS3.(15932, 15965)
    ABS4.(15351, 15932)
    ABS5.(15438, 15932)
    ABS6.(15438, 15525)
    ABS7.(15436, 15933)
    ABS8.(15438, 15933)
    ABS9.(15351, 15438)
    REL1.(15013, 15564)
    REL2.(15317, 15371)
    REL3.(15436, 15525)
    REL4.(15769, 15874)
    REL5.(15521, 15832)
    REL6.(15521, 15825)
    REL7.(15564, 15615)
    REL8.(15018, 15615) /



SEM(I)  Courses taught in three hour seminar format

   / 15311AC, 15311DF, 15311GI, 15311JL, 15962, 15099,
     15141, 15664, 15674, 15769 /
```

TABLE CONTRIB (I,J)   Faculty Preference for Time J

	T1	T2	T3	T4	T5	T6	T7	T8
15011AF	1	1	1	1	1	5	5	1
15011GL	1	1	1	1	1	5	5	1
15061AF	1	5	5	3	1	5	5	3
15061GL	1	5	5	3	1	5	5	3
15280AB	5	5	2	1	5	5	2	1
15280CD	5	4	3	1	5	4	3	1
15280EF	5	5	2	1	5	5	2	1
15280GH	5	4	3	1	5	4	3	1
15280IJ	4	5	4	4	1	1	1	1
15280KL	4	5	4	4	1	1	1	1
15311AC	1	1	1	5	1	1	1	5
15311DF	1	1	1	5	1	1	1	5
15311GI	1	1	1	5	1	1	1	5
15311JL	1	1	1	5	1	1	1	5
15515AD	3	3	3	3	3	3	3	3
15515EH	3	3	3	3	3	3	3	3
15515IL	3	3	3	3	3	3	3	3
15560AF	5	5	5	5	4	4	4	4
15560GL	5	5	5	5	4	4	4	4
15930AC	3	3	3	3	3	3	3	3
15930DF	5	4	1	1	5	4	1	1
15930GI	4	5	1	1	5	4	1	1
15930JL	1	5	4	1	1	3	4	1
15001	3	3	3	3	3	3	3	3
15013	3	5	3	5	1	1	1	1
15018	3	3	3	3	3	3	3	3
15034	1	1	1	1	5	1	1	1
15035	1	5	1	5	1	4	3	2
15053	2	5	3	4	2	5	2	4
15058	1	5	3	2	1	5	4	3
15059	5	5	1	3	3	3	1	1
15065	1	5	1	1	1	1	1	1
15075	3	3	3	3	3	3	3	3
15081	5	5	1	3	5	5	1	3
15083	3	3	3	3	3	3	3	3
15099	1	1	1	5	1	1	1	1
15141	1	1	1	1	1	1	1	5
15221	4	5	2	4	3	3	1	1
15231	3	3	3	3	3	3	3	3
15232	3	3	3	3	3	3	3	3
15301	1	5	1	1	1	1	1	1
15312	1	3	5	4	1	2	5	4
15317	3	3	3	3	3	3	3	3
15351	1	5	4	1	1	5	4	1
15361	2	5	4	4	2	5	4	4
15371	5	1	1	1	1	1	1	1
15412	3	3	3	3	3	3	3	3
15413	3	3	3	3	3	3	3	3
15415A	3	3	3	3	3	5	3	3
15415B	3	3	3	3	3	3	3	5
15435A	5	5	3	3	3	3	3	3
15435B	5	5	3	3	3	3	3	3
15436	4	5	2	4	3	3	1	1
15437	3	3	3	5	3	3	3	5
15438	1	1	5	5	1	1	5	5
15501A	1	4	5	3	1	4	5	4
15501B	1	4	5	3	1	4	5	4
15521	3	3	3	3	3	3	3	3
15525	3	3	3	3	3	3	3	3

```
     15539          3    3    3    3   3     3    3    3
     15564          5    4    1    1   5     4    1    1
     15565          5    4    3    2   5     4    3    2
     15568          3    2    1    5   1     1    1    1
     15601          1    1    1    5   1     1    1    1
     15615          3    3    3    3   3     3    3    3
     15664          1    3    5    5   1     3    1    5
*
*
*    15665    This course was commented out since it is included
*             with 15664 for modelling purposes.
*
*
     15671          1    1    3    5   1     1    1    4
     15674          1    1    1    4   1     1    1    5
     15691          1    1    1    1   1     1    5    1
     15763          3    3    3    3   3     3    3    3
     15768          5    4    3    2   3     2    2    1
     15769          1    1    1    5   1     1    1    3
     15812          1    1    1    1   2     5    5    4
     15814          3    3    3    3   3     3    3    3
     15824          2    5    5    3   1     4    4    3
     15825          4    5    4    3   2     3    2    2
     15832          1    1    1    1   2     5    5    4
     15874          3    3    5    3   3     3    3    3
     15878          3    3    3    5   3     3    3    3
     15932          3    3    3    3   3     3    3    3
     15933          3    3    3    3   3     3    3    3
     15951B         1    4    5    3   1     4    5    4
     15951C         4    5    4    4   1     1    1    1
     15962          1    1    1    5   1     1    1    1
     15964          3    3    3    3   3     3    3    3
     15965          1    5    4    1   1     5    4    1     ;


  TABLE COREHALF (CORE, HALF)  Mapping of CORE Courses to Half Semesters

                  H1     H2

     15011AF       1      1
     15011GL       1      1
     15061AF       1      1
     15061GL       1      1
     15280AB              1
     15280CD              1
     15280EF              1
     15280GH              1
     15280IJ              1
     15280KL              1
     15311AC       1      1
     15311DF       1      1
     15311GI       1      1
     15311JL       1      1
     15515AD       1      1
     15515EH       1      1
     15515IL       1      1
     15560AF       1
     15560GL       1
     15930AC              1
     15930DF              1
     15930GI              1
     15930JL              1   ;
```

```
TABLE COREMAP (CORE, S)  Mapping of CORE Sections to Courses

          SecA SecB SecC SecD SecE SecF SecG SecH SecI SecJ SecK SecL

15011AF     1    1    1    1    1    1
15011GL                                   1    1    1    1    1    1
15061AF     1    1    1    1    1    1
15061GL                                   1    1    1    1    1    1
15280AB     1    1
15280CD               1    1
15280EF                         1    1
15280GH                                   1    1
15280IJ                                             1    1
15280KL                                                       1    1
15311AC     1    1    1
15311DF                    1    1    1
15311GI                                   1    1    1
15311JL                                                  1    1    1
15515AD     1    1    1    1
15515EH                         1    1    1    1
15515IL                                             1    1    1    1
15560AF     1    1    1    1    1    1
15560GL                                   1    1    1    1    1    1
15930AC     1    1    1
15930DF                    1    1    1
15930GI                                   1    1    1
15930JL                                                  1    1    1  ;


Table Rooms(J,R)  Number of Rooms of type R available at time J

          R1     R2     R3     R4

     t1   1      4      6      9
     t2   1      3      6      9
     t3   1      3      6      9
     t4   1      4      6      9
     t5   1      3      6      9
     t6   1      3      6      9
     t7   1      4      6      9
     t8   1      4      6      9  ;

*  Extra room in R2 available when not used by Sloan Fellows or
*  when a joint Sloan Fellows/Masters course is offered.
*  See Text, Section VI.
*

 Parameter Enroll(I)     Estimated Enrollment for each course

 /
   15011AF    123
   15011GL    123
   15061AF    62
   15061GL    62
   15280AB    23
   15280CD    23
   15280EF    23
   15280GH    23
   15280IJ    23
   15280KL    23
   15311AC    41
   15311DF    41
```

	15311GI	41
	15311JL	41
	15515AD	58
	15515EH	58
	15515IL	58
	15560AF	70
	15560GL	70
	15930AC	50
	15930DF	50
	15930GI	50
	15930JL	50
	15001	53
	15013	19
	15018	61
	15034	14
	15035	30
	15053	46
	15058	46
	15059	18
	15065	30
	15075	95
	15081	34
	15083	19
	15099	6
	15141	2
	15221	83
	15231	20
	15232	30
	15301	112
	15312	41
	15317	20
	15351	50
	15361	35
	15371	54
	15412	49
	15413	30
	15415A	45
	15415B	45
	15435A	60
	15435B	60
	15436	95
	15437	68
	15438	92
	15501A	70
	15501B	70
	15521	12
	15525	13
	15539	48
	15564	61
	15565	42
	15568	64
	15601	49
	15615	49
	15664	60
*	15665	commented out since scheduled with 15664 for modelling purposes.
	15671	7
	15674	9
	15691	10
	15763	22
	15768	65
	15769	60
	15812	98
	15814	35
	15824	26
	15825	45

```
   15832       50
   15874       21
   15878       5
   15932       109
   15933       50
   15951B      80
   15951C      40
   15962       20
   15964       50
   15965       45
                                         /


 Parameter Roommap(I,R)  Mapping of Courses to Room size groups;

  Roommap(I,"R1") $  (Enroll(I) GE 90) = 1;
  Roommap(I,"R2") $ ((Enroll(I) GE 55) and (Enroll(I) LE 89)) = 1;
  Roommap(I,"R3") $ ((Enroll(I) GE 25) and (Enroll(I) LE 54)) = 1;
  Roommap(I,"R4") $  (Enroll(I) LE 24) = 1;


Variables X(I,J)              ASSIGNMENT OF COURSE I TO TIME J
          OBJ                 OBJECTIVE FUNCTION;


Positive Variables X;


* This section fixes some courses.  See Thesis Text.

X.FX("15768","t1") = 1;
X.FX("15769","t4") = 1;
X.FX("15311AC","t4") = 1;
X.FX("15311DF","t4") = 1;
X.FX("15311GI","t4") = 1;
X.FX("15311JL","t4") = 1;
X.FX("15035","t2") = 1;
X.FX("15075","t3") = 1;
X.FX("15061AF","t2") = 1;
X.FX("15061GL","t3") = 1;


Equations ASGT(I)             Each course must get a time
          ROOMCAP(J,R)        Room capacity for each room type and time
          REQCORE(S,HALF,J)   No CORE conflicts
          FACULTY(F,J)        No Faculty Conflicts
          CONCEN(CON,J)       No Concentration Conflicts
          CONPAIRS(P,J)       No conlficts for high overlap course pairs
          SEMINAR             Three Hour Seminars taught at end of day
          OFFMW(FMW,JTT)      Single day teaching for MW faculty
          OFFTT(FTT,JMW)      Single day teaching for TTH faculty

*   See thesis text (Section IV.A) for an explanation of the back to back constraints

          NBBMWAM(FMWNBB)     No back to back for Monday and Wednesday AM
          NBBMWPM(FMWNBB)     No back to back for Monday and Wednesday PM
          NBBTTAM(FTTNBB)     No back to back for Tuesday and Thursday AM
          NBBTTPM(FTTNBB)     No back to back for Tuesday and Thursday PM
          YBBMW1(FMWYBB)      Yes back to back for Mon and Wed eqtn 1
          YBBMW2(FMWYBB)      Yes back to back for Mon and Wed eqtn 2
          YBBMW3(FMWYBB)      Yes back to back for Mon and Wed eqtn 3
          YBBMW4(FMWYBB)      Yes back to back for Mon and Wed eqtn 4
```

```
*  These constraints commented out because FTTYBB was null

*          YBBTT1(FTTYBB)      Yes back to back for Tues and Thurs eqtn 1
*          YBBTT2(FTTYBB)      Yes back to back for Tues and Thurs eqtn 2
*          YBBTT3(FTTYBB)      Yes back to back for Tues and Thurs eqtn 3
*          YBBTT4(FTTYBB)      Yes back to back for Tues and Thurs eqtn 4
           OBJDEF              Definition of objective function;


  ASGT(I)..            SUM(J, X(I,J)) =E= 1;

  ROOMCAP(J,R)..       SUM(I, ROOMMAP(I,R)* X(I,J)) =L= ROOMS(J,R);

  REQCORE(S,HALF,J).. SUM(CORE,
                       COREHALF(CORE,HALF) * COREMAP(CORE,S) * X(CORE,J)) =L= 1;

  FACULTY(F,J)..       SUM(I $ FTOI(F,I), X(I,J)) =L= 1;

  CONCEN(CON,J)..      SUM(I $ CONTOI(CON,I), X(I,J)) =L= 1;

  CONPAIRS(P,J)..      SUM(I $ PTOI(P,I), X(I,J)) =L= 1;

  SEMINAR..            SUM((SEM,TSEM), X(SEM,TSEM)) =E= 0;

  OFFMW(FMW,JTT)..     SUM(I $ FTOI(FMW,I), X(I,JTT)) =E= 0;

  OFFTT(FTT,JMW)..     SUM(I $ FTOI(FTT,I), X(I,JMW)) =E= 0;

  NBBMWAM(FMWNBB)..    SUM(I $ FTOI(FMWNBB,I), (X(I,"T1") + X(I,"T2"))) =L= 1;

  NBBMWPM(FMWNBB)..    SUM(I $ FTOI(FMWNBB,I), (X(I,"T3") + X(I,"T4"))) =L= 1;

  NBBTTAM(FTTNBB)..    SUM(I $ FTOI(FTTNBB,I), (X(I,"T5") + X(I,"T6"))) =L= 1;

  NBBTTPM(FTTNBB)..    SUM(I $ FTOI(FTTNBB,I), (X(I,"T7") + X(I,"T8"))) =L= 1;

  YBBMW1(FMWYBB)..     SUM(I $ FTOI(FMWYBB,I), (X(I,"T1") + X(I,"T3"))) =L= 1;

  YBBMW2(FMWYBB)..     SUM(I $ FTOI(FMWYBB,I), (X(I,"T1") + X(I,"T4"))) =L= 1;

  YBBMW3(FMWYBB)..     SUM(I $ FTOI(FMWYBB,I), (X(I,"T2") + X(I,"T3"))) =L= 1;

  YBBMW4(FMWYBB)..     SUM(I $ FTOI(FMWYBB,I), (X(I,"T2") + X(I,"T4"))) =L= 1;

*  YBBTT1(FTTYBB)..     SUM(I $ FTOI(FTTYBB,I), (X(I,"T5") + X(I,"T7"))) =L= 1;

*  YBBTT2(FTTYBB)..     SUM(I $ FTOI(FTTYBB,I), (X(I,"T5") + X(I,"T8"))) =L= 1;

*  YBBTT3(FTTYBB)..     SUM(I $ FTOI(FTTYBB,I), (X(I,"T6") + X(I,"T7"))) =L= 1;

*  YBBTT4(FTTYBB)..     SUM(I $ FTOI(FTTYBB,I), (X(I,"T6") + X(I,"T8"))) =L= 1;

   OBJDEF..             OBJ =E= SUM((I,J), CONTRIB(I,J)*X(I,J));

 MODEL SCHED FIRST PASS / ALL / ;

 SOLVE SCHED USING LP MAXIMIZING OBJ;

 Display X.L;
 DISPLAY REQCORE.L;
 DISPLAY FACULTY.L;
 DISPLAY CONCEN.L;
```

REFERENCES

Akkoyunlu, E.A. (1971). "Linear Characterization of the Solutions of the Quadratic Assignment Problem." Available as TR-4, Department of Computer Science, SUNY at Stony Brook, New York.

Akkoyunlu, E.A. (1973). "A Linear Algorithm for Computing the Optimum University Timetable." The Computer Journal, Vol. 16, No. 4 (November), pp. 347-350.

Barham, Alan M. and John B. Westwood. (1978). "A Simple Heuristic to Facilitate Course Timetabling." Journal of the Operational Research Society, Vol. 29, No. 11 (November), pp. 1055-1060.

Bradley, Stephen P., Arnoldo C. Hax, and Thomas L. Magnanti. (1977). Applied Mathematical Programming. Reading, MA: Addison-Wesley Publishing Company.

Brooks, R. and A. Geoffrion. (1966). "Finding Everett's Lagrange Multipliers by Linear Programming." Operations Research, Vol. 14, No. 6 (November-December), pp. 1149-1153.

de Werra, D. (1985). "An Introduction to Timetabling." European Journal of Operational Research, Vol. 19, No. 2 (February), pp. 151-162.

Dyer, James S. and John M. Mulvey. (1976). "An Integrated Optimization/Information System for Academic Departmental Planning." Management Science, Vol. 22, No. 12 (August), pp. 1332-1341.

Findlay, William and David A. Watt. (1978). Pascal, An Introduction to Methodical Programming. Potomac, MD: Computer Science Press, Inc.

Fisher, Marshall L. (1972). "Optimal Solution of Scheduling Problems Using Lagrange Multipliers: Part II." In: Symposium on the Theory of Scheduling and its Applications, North Carolina State University, 1972. Berlin: Springer-Verlag, 1972.

Fisher, Marshall L. (1973). "Optimal Solution of Scheduling Problems Using Lagrange Multipliers: Part I." Operations Research, Vol. 21, No. 5 (September-October), pp. 1114-1127.

Fisher, Marshall L. (1981). "The Lagrangian Relaxation Method for Solving Integer Programming Problems." Management Science, Vol. 27, No. 1 (January), pp. 1-18.

Fisher, Marshall L. (1985). "An Applications Oriented Guide to Lagrangian Relaxation." Interfaces, Vol. 15, No. 2 (March-April), pp. 10-21.

Fisher, M.L., W.D. Northrup, and J.F. Shapiro. (1975). "Using Duality to Solve Discrete Optimization Problems: Theory and Computational Experience." Mathematical Programming Study, 3, pp. 56-94.

Geoffrion, A. M. (1974). "Lagrangean Relaxation for Integer Programming." Mathematical Programming Study, 2, pp. 82-114.

Glassey, C. Roger and Michael Mizrach. (1986). "A Decision Support System for Assigning Classes to Rooms." Interfaces, Vol. 16, No. 5 (September-October), pp. 92-100.

Graham, Daniel A. (1980). Microeconomics. Lexington, MA: D.C. Heath and Company.

Held, Michael H., Philip Wolfe, and Harlan D. Crowder. (1974). "Validation of Subgradient Optimization." Mathematical Programming, Vol. 6, No. 1 (February), pp. 62-88.

Kendrick, David and Alexander Meeraus. (1985). GAMS: An Introduction. Washington, DC: The World Bank, Development Research Department.

McClure, Richard H. and Charles E. Wells. (1986). "Departmental Planning in Academia by Decision Support." College and University, Vol. 61, No. 2 (Winter), pp. 81-89.

Mulvey, John M. (1982). "A Classroom/Time Assignment Model." European Journal of Operational Research, Vol. 9, No. 1 (January), pp. 64-70.

Prime Computer, Inc. (1980). Prime User's Guide IDR 4130. Framingham, MA: Prime Computer, Inc.

Romero, Bernardo Prida. (1982). "Examination Scheduling in A Large Engineering School: A Computer-Assisted Procedure." Interfaces, Vol. 12, No. 2 (April), pp. 17-23.

Sabin, G. C. W. and G. K. Winter. (1986). "The Impact of Automated Timetabling on Universities -- A Case Study." Journal of the Operational Research Society, Vol. 37, No. 7, pp. 689-693.

Schmidt, G. and T. Strohlein. (1979). "Timetable Construction: An Annotated Bibiliography." The Computer Journal, Vol. 23, No. 4 (November), pp. 307-316.

Shapiro, Jeremy F. (1971). "Generalized Lagrange Multipliers in Integer Programming." Operations Research, Vol. 19, No. 1 (January-February), pp. 68-76.

Silver, Edward A., R. Victor Vidal, and Dominique de Werra. (1980). "A Tutorial on Heuristic Methods." European Journal of Operational Research, Vol. 5, No. 3 (September), pp. 153-162.

Tripathy, Arabinda. (1980). "A Lagrangean Relaxation Approach to Course Timetabling." Journal of the Operational Research Society, Vol. 31, No. 7 (July), pp. 599-603.

Tripathy, Arabinda. (1984). "School Timetabling -- A Case in Large Binary Integer Linear Programming." Management Science, Vol. 30, No. 12 (December), pp. 1473-1489.