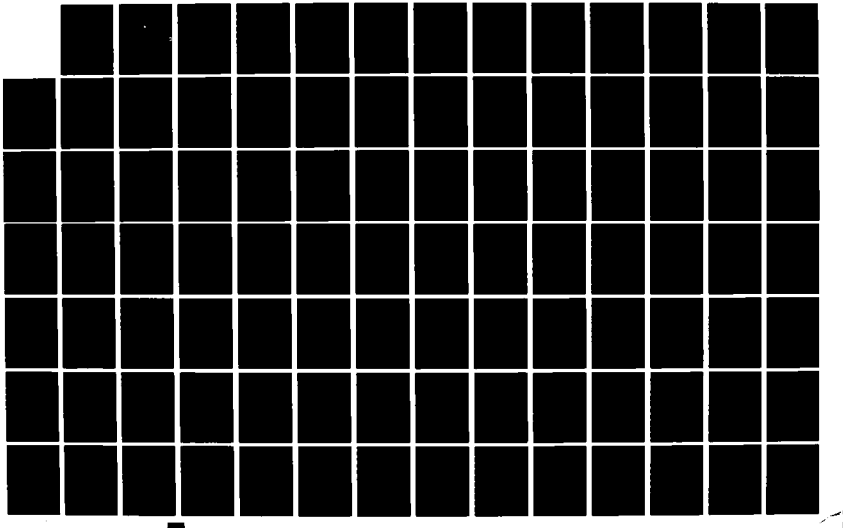
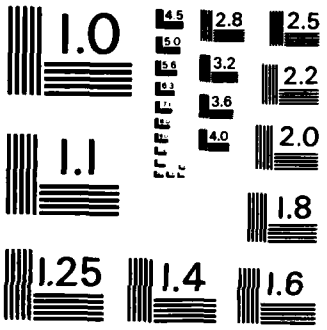
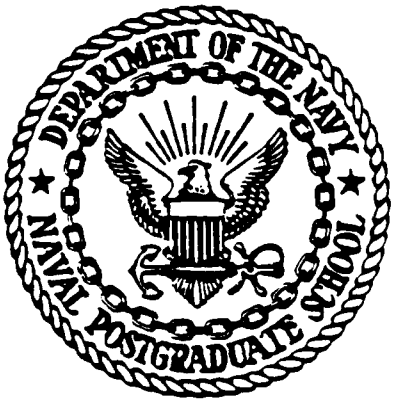
AD-A161 396 THE NAVAL POSTGRADUATE SCHOOL SCHEDULING SYSTEM: A HEURISTIC APPROACH(U) NAVAL POSTGRADUATE SCHOOL MONTEREY CA D W FIEGAS SEP 85 1/2

UNCLASSIFIED F/G 12/1 NL





# NAVAL POSTGRADUATE SCHOOL
## Monterey, California






# THESIS

THE NAVAL POSTGRADUATE SCHOOL SCHEDULING SYSTEM: A HEURISTIC APPROACH

by

Dietmar W. Fiegas

September 1985

Thesis Advisor: R. Kevin Wood

Approved for public release; distribution is unlimited.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

| REPORT DOCUMENTATION PAGE | | READ INSTRUCTIONS BEFORE COMPLETING FORM |
|---|---|---|
| 1. REPORT NUMBER | 2. GOVT ACCESSION NO. AD-A161396 | 3. RECIPIENT'S CATALOG NUMBER |
| 4. TITLE (and Subtitle) The Naval Postgraduate School Scheduling System: A Heuristic Approach | | 5. TYPE OF REPORT & PERIOD COVERED Master's Thesis September, 1985 |
| | | 6. PERFORMING ORG. REPORT NUMBER |
| 7. AUTHOR(s) Dietmar W. Fiegas | | 8. CONTRACT OR GRANT NUMBER(s) |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, CA 93943-5100 | | 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS |
| 11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, CA 93943-5100 | | 12. REPORT DATE September, 1985 |
| | | 13. NUMBER OF PAGES 139 |
| 14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) | | 15. SECURITY CLASS. (of this report) |
| | | 15a. DECLASSIFICATION/ DOWNGRADING SCHEDULE |

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

timetable, time table, time-table, College Schedules, University Schedules, Computer Science, Examination Schedules, Operations Research, Heuristics, Scheduling, Integer Linear Program

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A heuristic approach is presented to solve the Naval Postgraduate School's quarterly scheduling problem for academic courses and final examinations. The current scheduling system is studied and an automated system for data collection is developed and implemented. An

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

S/N 0102-LF-014-6601



SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

automated system for the scheduling of final examinations is designed and implemented. Results using real data from one quarter produced feasible solutions to the final examination scheduling problem of 1700 students, 267 courses and 850 sections. The academic course scheduling heuristic is discussed including an integer linear programming approach to the timetabling and distribution problem of students among segments of the same course. An example with an optimal solution to the single course scheduling problem is presented.

Approved for public release; distribution is unlimited.

# The Naval Postgraduate School Scheduling System: A Heuristic Approach

by

Dietmar W. Fiegas
Captain, German Army
Dipl. Ing., Fachhochschule des Heeres Darmstadt, 1977

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
September 1985

Author: _____
Dietmar W. Fiegas

Approved by: _____
R. Kevin Wood, Thesis Advisor

_____
Richard E. Rosenthal, Second Reader

_____
Alan R. Washburn, Chairman,
Department of Operations Research

_____
Kneale T. Marshall,
Dean of Information and Policy Sciences

## ABSTRACT

A heuristic approach is presented to solve the Naval Postgraduate School's quarterly scheduling problem for academic courses and final examinations. The current scheduling system is studied and an automated system for data collection is developed and implemented. An automated system for the scheduling of final examinations is designed and implemented. Results using real data from one quarter produced feasible solutions to the final examination scheduling problem of 1700 students, 267 courses and 850 sections. The academic course scheduling heuristic is discussed including an integer linear programming approach to the timetabling and distribution problem of students among segments of the same course. An example with an optimal solution to the single course scheduling problem is presented.

## THESIS DISCLAIMER

The reader is cautioned that computer programs developed in this research may not have been exercised for all cases of interest. While every effort has been made, within the time available, to ensure that the programs are free of computational and logic errors, they cannot be considered validated. Any application of these programs without additional verification is at the risk of the user.

## TABLE OF CONTENTS

## LIST OF TABLES

## LIST OF FIGURES

## ACKNOWLEDGEMENT

This work would not have been possible without the full support of the scheduler, Mary Horn, who introduced the scheduling system to me and answered all questions to the process patiently. My thesis advisor, Professor Kevin Wood, kept my mind in the right direction during the research. Hans Dolnan assisted in the building of the FOCUS files and contributed valuable suggestions. The type setting was assisted by Larry Frazier. My wife Ursula backed the work generously although suffering under my involvement in the topic and the little time I was able to devote to her.

I would like to thank all of them very much for their support and encouragement.

# I. INTRODUCTION

## A. BACKGROUND

The US Naval Postgraduate School (NPS) in Monterey is an institution dedicated to the postgraduate education of Naval Officers and Officers of the other services as well as officers from allied countries. The purpose of this thesis is to develop a method to schedule the academic courses and final examinations of an academic period of this school.

Most students are assigned to this school in order to achieve a Master's Degree in one of the offered curricular programs. As all the students receive their full salary and no fees are paid, the student is expected to devote his entire time to the achievement of this goal.

The special situation of school and students leads to a fairly special academic cycle: Between 1500 and 2000 students study four quarters a year. A quarter consists of eleven weeks of academic work and one week of final examinations. There is a two week break in summer and over Christmas and New Year.

During a particular quarter, there are about 300 courses taught and there are between 800 and 1000 distinct parallel sequences to be scheduled.

All sorts of other constraints in addition to the many different sequences and the short time span of 12 weeks, makes the scheduling process difficult. Although parts of the scheduling process are computer-aided, and despite earlier attempts to computerize it completely, the schedule is currently manually produced.

## B. PROBLEM AND MOTIVATION

Every quarter NPS faces the problem of creating the normal academic schedule and the final examination schedule. The normal academic schedule can be stated:

**Schedule all required courses**

subject to the constraints:

- Every student must be abl. to attend all courses of his/her approved sequence.
- Assigned rooms must be of the right size, suitable for the course purpose and in a designated area of the NPS campus.
- The faculty member who is assigned to teach the particular course must be available at the period.
- Every course must be scheduled with as many periods per week as indicated in the NPS catalog and must be scheduled with the appropriate number of periods per teaching block.
- A variety of other constraints imposed by the administration and the faculty must be observed.

The objective and the constraints for the final examination schedule are:

**Schedule examinations for all courses within four consecutive days of one week**

subject to the constraints:

- All courses have a two hour block of examination time.
- All segments of one course must have the exam during the same period.
- There are at most two exams for each student per day.
- The assigned rooms must be large enough for the number of students taking the exam in that room.
- A course may have more than one room for the exam but all rooms must be on the same floor of one building.
- Other constraints imposed by the teaching faculty member.

On a year-round basis, one full-time scheduler is working on the building of those two schedules. During the actual scheduling time, between the sixth and eleventh week of each quarter, the work is done by two persons. All updating of files in the registrar's office is stopped during this period to ensure undisturbed building of the schedules.

The manual scheduling of a complex system like this involves a long training of the scheduler to be able to produce acceptable results. A detailed knowledge of the operations of the school and its departments is also essential. This same person has to sort thousands of data cards over and over again during the scheduling process, fill in assigned hours and room numbers and erase data in the trial and error process of manual scheduling.

These two factors, the standstill in the registrar operation and the extremely high percentage of manual work during the actual scheduling, leads to the idea of computerizing the scheduling process and thus decreasing the necessary time and amount of manual work.

## C. THE APPROACH TO THE PROBLEM

Two approaches to solving the scheduling problem seem possible: a mathematical programming approach or a heuristic solution technique. To determine which approach to take, previous experience with the matter and reported results from the open literature were considered.

### 1. *Previous Experience*

There were earlier attempts at NPS to computerize the scheduling process. In 1966 HAMS [Ref. 1], the ***Heuristic Academic Master Scheduler*** was created at NPS. It was written in the assembly language SCRAP and ran on the school's IBM 1604. HAMS tried to imitate the manual scheduling work and produced results comparable to those of the human scheduler during trial runs.

There were several reasons why this project was abandoned and why this thesis does not continue where HAMS stopped:

- The HAMS system was unable to schedule about 5% to 10% of all courses. It turned out, that after a HAMS scheduling run, courses were scheduled such that there was hardly a period or room left for the scheduling of the remaining courses. Due to the algorithm used, it was impossible to schedule the remaining courses without changing the whole schedule. Therefore, the system was potentially useless and the system was never used by the scheduler.

- The mainframe has been replaced twice since HAMS was programmed and the old programs are difficult to interpret.
- A major portion of the old system (assigning flight schedules to aviator students) is no longer relevant.

There are no further known attempts to computerize the scheduling process at NPS.

## 2. *Literature*

Since the late 1950s and peaking in the 1960s, the problem of **timetabling** an expression widely use for course scheduling, has been appearing in the literature. Due to its complexity and large size interest in the problem has increased with more common availability of computers with large memories in recent years.

The work done on timetabling can be subdivided into two categories, as Tripathy **[Ref. 2]** does: **School timetabling problems** and **College/University timetabling.** School timetabling in this sense is considered to be harder problem than college/university timetabling due to the fact that conflicts, resulting from interactions between departments are less likely in the latter.

### a. School Timetabling

A technical history by Dempster, Lethbridge and Ulph, **[Ref. 3]**, tracks the evolution of computer timetabling in the United Kingdom through 1975 and shows the typical dimensions of actual school timetabling systems. All the early systems were designed to aid the actual construction of the timetable after basic requirements (curricula) were set by the school staff. Dempster et al mention one major factor affecting timetabling and its ability to achieve satisfactory schedules: Besides explicit requirements (constraints) there exists a large number of implicit ones. These are only known vaguely and appear in real time during the scheduling process. The human scheduler, a senior staff member in these school systems, decides all these upcoming questions on his/her own out of his/her experience and knowledge of the internal workings of the school. Dempster et al stress the fact that it is

impossible to specify the requirements completely and that systems should therefore allow the scheduler to interact closely with the computer in timetable construction.

After listing the requirements a computer timetabling system should be able to handle, Dempster et al look at three generations of actual systems. The first generation of systems in the early 1960s was based on heuristics only and solved small sized problems, e.g. the Teddington System: 35 periods, 26 classes, 90 min CPU time. The systems of this generation were:

- Teddington System
- Ontario School Scheduling Programme
- New Castle System

The length of time needed to solve even an unrealistically small problem was the reason that these systems never left experimental stages.

Second generation systems produced for the first time reasonable results using the ideas of their unsuccessful predecessors. Similar, but improved heuristics were used and the systems were run on more sophisticated computers. Systems of this generation were used in real applications in schools in Great Britain. Second generation systems are:

- Birmingham System
- Nor-Data System
- New Zealand System
- Price-Waterhouse
- Scicon System

Third generation/systems differ in the fact that they start earlier in the school scheduling process. During the curriculum planning phase, an outline timetable was produced by the Strathclyde System. Outgoing from this, the actual timetable was constructed later. For the first part, integer linear programming was used. Besides sometimes unacceptably long run times, solutions could not always be produced due to infeasibilities in the large constraint set.

The Oxford System, the other third generation system, is the first system where an interactive approach is tried. The system

starts by using a feasible start timetable, usually parts of a previous scheduling period's timetable. Feasiblity is required only for certain major requirements. Dempster et al mention that one package was operational and that yet another one was still in development. The later one was said to use interactive refinement.

Dowsland and Lim [Ref. 4], mention that the Oxford System is operational but that none of the existing systems produces 100% complete results. A comparison of the Nor-Data System, the New Zealand System and the Oxford System concludes that the Nor-Data System gave the best results.

b. College/University Timetabling

Tripathy [Ref. 2] and [Ref. 5] describes College/University timetabling problems. The problem is defined as the scheduling of 900 subjects over a year. Students are enrolled in 25 specializations. Students from 4 to 6 different streams study particular subjects together. The academic year has three terms. Separate time tables for each term are constructed, where a term time table has about 400 to 450 subjects. A week consists of 30 periods.

To be able to solve the problem as an integer linear program (ILP), the number of variables and constraints is reduced by grouping of students into student groups and subjects into subject groups.

The problem formulation is divided into three stages:

i) Determine the group of students which is to attend a particular subject,

ii) Specify the facility required by the subject,

iii) Determine when each subject is to be taken.

In Tripathy's approach, (i) and (ii) are done manually. Stage (iii) is formulated as an ILP. The objective function maximizes the desirability of the assignment of subject group i to period j.

The constraints take care of the total number of periods per subject per week, the availability of rooms out of a set of room groups and the fact that a student can only attend one subject group during one period.

A lagrangian relaxation method is used as a solution technique. Computational results are reported for the described case with 33 student groups, 84 subject groups and 5 room groups, where single and double period classes were scheduled.

Another author concerned with university timetabling is Selim, with an article about university faculty timetabling [Ref. 6] and course and lecture timetables [Ref. 7]. Both systems are based on heuristic solution techniques close to those of Almond [Ref. 8] but fail to schedule about 15% of all courses and do not consider room assignments.

c. Examination Timetable

There is some literature about examination timetabling in general and also about final examination timetabling. Romero [Ref. 9] describes a system which schedules examinations and assigns rooms with interaction to "highlight and clarify references in making decisions and constraints imposed by previous decisions." All exams have to be written within a time frame, each course offers up to three possible examination periods, some specific class rooms are needed, and different exams for one student should be separated by some minimum period of time. There is some course interaction, but the main student body is enrolled in noninteracting courses. A heuristic algorithm assigns rooms and dates. In case of difficulties, a human decision maker has to give inputs.

In contrast to the very simple system described above, Mehta [Ref. 10] reports on a more complex procedure using a vertex coloring method for building final examination schedules. The problem is to schedule final examinations over a short time period, subject to the constraint that some exams cannot take place during the same time and that all the courses taken by a student need to be scheduled at different times.

A network is created where nodes represent courses and arcs between nodes represent a conflict between two courses. Then, a graph coloring method is used to find the minimum number of

independent course sets. Room problems were incorporated into the network. After deriving an acceptable schedule by this method, an adjustment phase is entered where time-frames are selected for exams of the same color. This is manually done by the registrar.

### 3. *Comparison to the Scheduling Problem at NPS*

The kind of problem a school timetabling system solves has only slight similarities with the problem at NPS. Schools face problems with a large number of students of the same grade to be distributed evenly among parallel classes. Once this problem is solved, a switching of some students from one class to another among the same grade is not very common and switching between grades practically nonexistent. Thus, a schedule can first be constructed and then students can be distributed according to the schedule. It is also possible to take the schedule of a previous year and transform it into a new feasible schedule. All this is not the case at NPS. The reported results of less than 100% success for all described systems are another reason not to consider such a system for use at NPS. A major factor for the acceptance of a scheduling system is its reliability and accuracy. Experience indicates that partial, infeasible schedules are of little practical value; the problem of scheduling the then remaining conflicting courses manually is, in most cases, as hard and time-consuming as the original problem itself.

The College/University timetabling problem described and solved by Tripathy seems to have enough similarities with the problem faced by NPS to be considered as a possible candidate. When comparing the particular data though, some differences emerge:

The definition of a student group is close to sections in the NPS system, but 33 student groups are one order of magnitude less than the number of sections at NPS. Tripathy defines a room group as the set of rooms of similar size and use. For the lecture rooms at NPS it would also be possible to form about 5 such groups. In addition to these five groups there are about 20 single use rooms where each builds a group by itself. These are lab rooms, terminal rooms, vault

rooms and rooms equiped with special features. Still another problem would cause a subdivision of the five lecture room groups: It is policy at NPS to schedule courses not anywhere on campus but in designated areas, depending on the department offering the course. All this leads to about 40 room groups.

Given that the number of subject groups are of the same size in both systems and that all the other constraints could be combined into the penalty cost of the objective function, the increase in room groups and student groups would increase the number of variables of a ILP formulation to a impractical number, on the order of $10^7$. An ILP to the single course problem, however, can be formulated and is found in Chapter V.B.5 of this thesis.

Given the drawbacks of the optimization systems described above, a heuristic scheme for scheduling seems more appropriate. This system would include interactive features to combine the human scheduler's knowledge of the school system with the speed and accuracy of a computer. A decision was made to built a heuristic scheduling system with the goal to achieve feasible solutions and to be able to handle the special problems at NPS.

For the examination schedule, Mehta seems to present a suitable way to solve parts of the problem. The situation at NPS, however, is much more constrained. A schedule with more than two exams for one student on any particular day is already infeasible; rooms are a scarce resource; and the overall time frame of four days is not negotiable. Therefore, for this problem too, none of the existing algorithms are appropriate. An alternate heuristic method is devised to incorporate all the specific problems existing at NPS.

## II. PROBLEM DESCRIPTION

To design a scheduling system which will be accepted by the user, it is necessary to achieve results qualitatively comparable or better than those currently possible. As the scheduling of the courses is imbedded into the much wider field of academic administration at NPS, the new system must have interfaces to existing operations. The change to the new system must be possible without disturbing or stopping the administrative process. Furthermore, the interaction between the human scheduler and the computerized system is a way of fitting the system to the continuous changes within the operations at NPS. All these requirements make it necessary to keep as much as possible of the current operation unchanged. Detailed knowledge of the current system, its terminology, data flow and data structures and the way the schedule is created currently is therefore essential.

### A. PROBLEM DEFINITIONS

#### 1. *Horizon*

For each quarter, a separate schedule is to be constructed. During the first eleven weeks, the schedule is the same every week. The last (twelfth) week is dedicated to final examinations only. Hence, there are two schedules to be constructed for a particular quarter:

i) The academic schedule; frame one week, Monday through Friday

ii) The final examination schedule; frame four days, Monday through Thursday

#### 2. *Lecture and Laboratory Hours*

One teaching or examination unit has a duration of 50 minutes followed by a ten minute break. There are nine such units in every working day. The majority of all courses are taught one hour per day

although there are other blocks consisting of two or three hours in a row, especially for laboratory-type courses ("labs").

3. *Academic Department*

The teaching function is performed by the faculty members of eleven academic departments and three interdisciplinary groups, in short, called "departments."

4. *Faculty*

The teaching is done by civilian and military faculty members of the Naval Postgraduate School; few courses are taught by guest professors. The assignment of a faculty member to a course and segment occurs within the departments. A load of two to three segments is average for those quarters when a faculty member is assigned to teach. Faculty members are involved in research projects, advising of master's or doctoral students and work in other sponsored areas. A number of constraints within the scheduling process are imposed by these additional activities.

5. *Curricular Office*

Students are grouped into different curricula groups. Students in one group pursue similar or closely related curricula.

6. *Course*

Courses offered by the departments are taught between 2 and 6 hours per week and consist of lecture and/or lab parts depending on the current course catalog [Ref. 11]. The number of lecture and lab hours per week are defined in this catalog. Some of the courses do not need schedules or class rooms.

7. *Segment*

Depending on the number of students enrolled in one particular course, it can be decided to split the course into, usually, two to four "segments." These may be taught by one or more faculty members.

The building of segments sometimes occurs only for the lecture or lab part of a course whereas the other part is taught as a whole. The department decides on the number of segments per course and the faculty member teaching a particular segment.

8. ***Section***

All students to be scheduled for the same sequence of courses in a particular quarter are grouped together into a "section." In almost all cases this will be the smallest unit to be scheduled individually.

9. ***Student***

Students are grouped into the same section if and only if they have the same sequence of courses in a quarter. There are required and elective courses. Guided by the curricular offices and sponsors, constrained by the offered courses and degree requirements, the student is free to choose among a certain subset of courses. A load of three to six courses is normal with four courses being the average.

10. ***Room***

Rooms for teaching of lecture and lab courses are in six buildings on campus. A total of 110 rooms with capacities from 10 to 80 people are available. Courses offered by a particular department are usually taught within a specified area of the campus. Vault rooms, rooms with computer terminals and specially equiped lab rooms are specifically requested by the teaching faculty member.

11. ***Meetings, Guest Lectures, Seminars and Refresher Courses***

There are regularly scheduled meetings within the departments, seminars for groups of students of certain curricula and guest lectures for the whole student body or some subsets. Times, duration and attending members are provided ahead of the scheduling period. Refresher courses for new students are taught every quarter from week seven through week twelve without scheduled final exams.

## B. CONSTRAINTS

The difficulty of scheduling final examinations and normal academic courses originates from many constraining factors. Some of the constraints are official policy; others are dictated by practical considerations and are not explicit in official policy. The latter have to be analyzed more intensivly and defined before a heuristic can be designed. This section describes the scheduling constraints.

### 1. *Final Examination Schedule*

The basic unit for a final exam is two contiguous hours. There are at most two exams for every section to be scheduled on any particular day of the final examination week.

a) In order to provide enough space for every student, a factor of 1.5 times the size of a segment is a guideline for selecting rooms. In order to reach this goal, the segments may be scheduled into more than one room on the same floor of one building. Several segments of one course taught by one faculty member may be scheduled in one room. All segments of a course must have the exam at the same time.

b) There are final exams in every course with a lecture part and no final exams for lab-only courses. This may be changed by the teaching faculty member.

c) All lecture rooms are available for exams during the final examination week, except rooms occupied by refresher courses.

d) There are preferred areas for rooms to be used for exams as can be seen in Table 1.

e) Every quarter there are some requests for early final examinations. They are scheduled during the first day if possible.

### 2. *Academic Schedule*

#### a. The standard setting

Without requests of any kind, the standard setting for an academic schedule is described as follows:

TABLE 1

COURSES AND THEIR PREFERRED ROOMS ON CAMPUS

| Course Indicator | Building | Floor |
|---|---|---|
| AE, ME | Halligan | |
| AS, CO, CM, IS, MN | Ingersol | 2, 3 |
| EE | Bullard<br>Spannagel | 2, 3, 4 |
| GH | 224 | |
| CS, MS, CS | Spannagel | 2, 3, 4 |
| MA | Ingersol | 2, 3 |
| OA, OS | Ingersol<br>Root | 3<br>2, left |
| MR | Root | 2, left |
| NS, ST, OC | Root | 2 right |
| PH, CC | Spannagel | 1, 2 |

i) Every section is scheduled in such a manner that all courses requested by a student can be taken. If no feasible schedule results out of this restriction, unusual sequences are sorted out and restructured by the curricular office and the students of the section.

ii) No more than one hour per course per day.

iii) The section is scheduled in a room as small as possible.

iv) Courses are scheduled in prefered areas, the same as with scheduling of final exams (Table 1).

v) There will be no scheduled course hours during regularly scheduled guest lectures, seminars etc. for the student or faculty members involved.

vi) There will be a lunch hour for every student or faculty member in the time period between 11 A.M. and 2 P.M.

vii) One segment per course is assumed if not indicated otherwise.

b. *Frequent requests constraining the scheduling process*

The most frequent constraints occuring every quarter are as follows:

i) The course must be divided into segments.

ii) Particular rooms are needed. The reason can be the need for lab equipment, higher than normal security, computer terminals, special teaching aids, etc.

iii) There are courses which need no class room

iv) The course/segment is taught in other than one-hour blocks. Two or three hour blocks are frequent requests.

v) A particular group of sections must be in one segment.

vi) The faculty member prefers to have or must have particular hours of the day or days of the week for the teaching of the segment.

vii) Faculty members like to have an hour between two teaching sessions.

viii) Faculty members participate in a course as students.

ix) Some courses are taught in a team of two or more faculty members.

x) There are actually more or less hours taught than in the current NPS catalog indicated.

xi) A segment of a course with lecture and lab parts is subdivided for the lab part.

xii) There are accelerated and refresher courses which are taught during the first or last 6 weeks of the quarter only. No final examination is scheduled for those courses.

xiii) Some courses are taught together during the same period in the same room.

## C. COLLECTION OF THE CONSTRAINT DATA

Some of the data needed for the scheduling process are provided in the form of computer files; not so the changing set of constraints. The current method of collecting the constraints is described and a faster computerized system is proposed in this section.

## 1. *The Current System*

The current method of collecting all the data about constraints is to write them on the pages of the department list printout. One of the existing computerized data files is the department list.

This list contains data about the courses: course number, participating sections, number of hours for lecture and lab per week and number of students, sorted by department. The inputs for the list are provided by the different curricula offices and departments.

The list is handed to the scheduler at the beginning of the scheduling phase. In order to collect the remaining data, the scheduler distributes the relevant parts of the list to the respective departments.

At the department level, all the constraints, as well as the teaching faculty member and the number of segments for each course are handwritten on the computer printout of the department list. There is no common form of providing the constraints. This leads to different degrees of detail and leaves much detail to the memory and knowledge of the scheduler. The process of distributing the department list to the departments and sending them back takes about three days.

The transformation of the constraint data from the handwritten form to a acceptable form for computational use takes about 30 man hours and the quality depends very much on a very good understanding of the scheduling process.

## 2. *An Automated System*

The analysis of previous department list described above resulted in a list of common constraints which is laid out in the form of a computer questionnaire. The questionnaire is written in FOCUS, a information control system in use by the registrar data management group. The program has been implemented and is easy to use by minimally trained personnel. The main advantage of this kind of questionnaire is the discretizing of the different constraints leading to a uniformity of answers which is necessary for computer use.

The data on the constraints for one quarter were typed in, by use of the program; one record for each faculty member and

segment. Among all input, only 15% of all records had constraints that did not fit one of the provided questions. For these cases, three lines for comments were provided. These "comment constraints" were very unusual requests, each by only one or two faculty members which most probably will not reoccur in the same form again.

The output file of this FOCUS program was used as input into the scheduling program.

To make the scheduling process faster, the constraint data should be provided via this or a similar program directly by the producers of the constraints, the departments. The different department-constraint files could then be easily connected. In addition to savings in time, a reduction in errors would be another advantage of this approach.

Appendix E through H provide a copy of the questionnaire screen, the FOCUS program listings, a small user's manual for the proposed method of usage and the MASTER file.

# III. COURSE AND SECTION INTERACTION

Every quarter, about 1700 students choose three to six courses from among the 250 to 300 courses offered. The relative freedom in the selection of courses leads to more than 800 sections, a section-to-student ratio of about 1.8 to 1. Define a **connection** between two courses to exist if and only if there is at least one section taking both courses. An analysis of the set of courses with respect to the degree of connections among them is contents of this chapter.

## A. SUBSETS OF THE COURSE-SECTION SET

The question arises whether all the courses are connected to each other, at least indirectly, as the high section-to-course ratio suggests. As students belong to different curricula, it might be possible to break up the set of courses into noninteracting subsets. The solution to this question has a great impact on the difficulty of the scheduling problem. If noninteracting subsets can be found, a solution technique would be possible where each subset is scheduled by itself. In the extreme case of many subsets of comparable size, the individual problems would be small enough to find an optimal solution with respect to an as yet to be determined measure of effectiveness. The other extreme of one big connected set implies that mathematical programming techniques will not be practical due to the size of the resulting problem. The number of feasible solutions, if any, is likely to be much smaller in this second case.

To find independent subsets an undirected network is set up where nodes correspond to courses and arcs to connections. The courses, originally named by two letters, followed by a four digit number, are renumbered with consecutive integers, starting from 1, according to their place in the lexicographically ordered set of courses. A graphical representation of a small part of the network is shown by Figure 3.1.

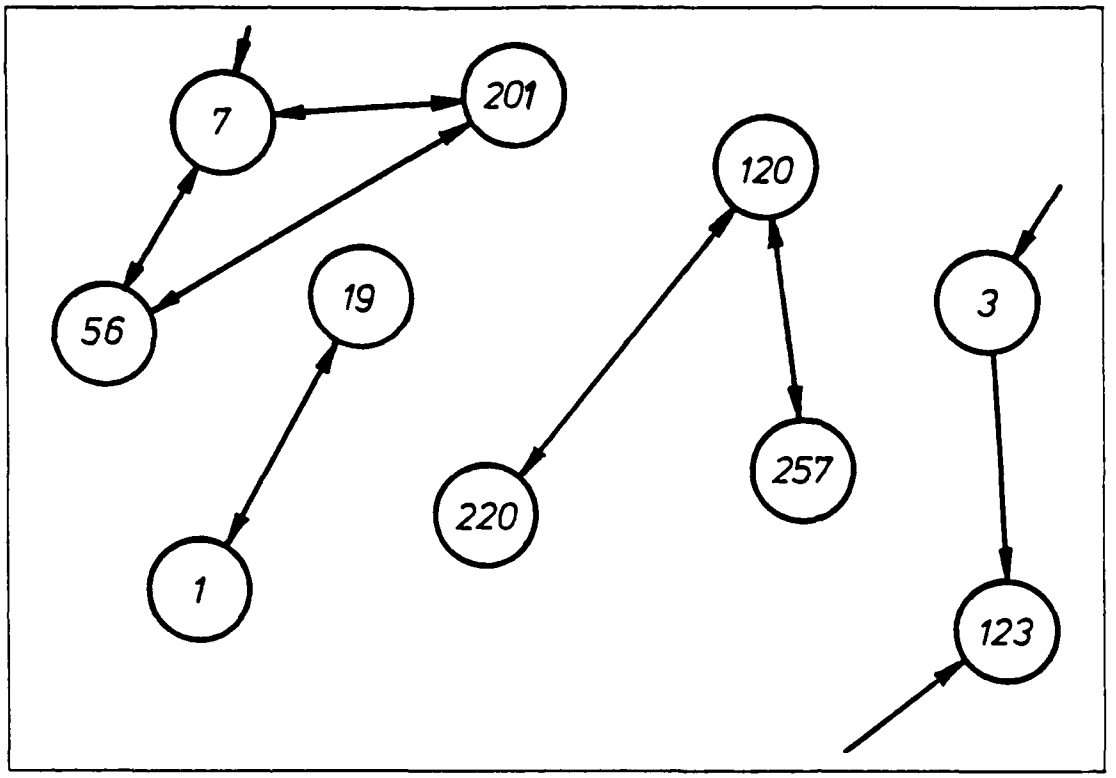



Figure 3.1 Network Representation of the Course/Section Set.

The task then is to find nodes like 1 and 19, which are connected with each other but which have no connection to the set {120, 220, 257} and thus form an independent set.

To find a solution, a breadth-first search routine can be employed:

1. Arbitrarily choose node i as starting node.
2. Find a path from the distinct node i to every other node j of the network. All connected nodes to node i, together with node i, form a partition of the network and thus create an independent subset.
3. Among the nodes not yet a member of one of the subsets formed by step 2 choose a new start node i and go to step 2.
4. Stop when all nodes belong to an independent subset or when only one node is left, forming a subset by itself.

Results using actual data from the third quarter 1985 showed that all courses were connected, except for courses not really belonging to the masters program like refresher and language courses. Hence, the scheduling process can not be broken up into independent pieces. This reinforces the validity of the decision to use a heuristic solution technique to find a feasible schedule.

## B. SAME - HOUR INTERACTION

To be able to build a schedule, one has to find all the courses which can be scheduled at the same time without interference. Figure 3.1 shows that either course 1 or course 19 can be scheduled at the same time with any other course of the set but not both since there is at least one section attending both courses.

To find a lower bound on hours per day necessary to schedule all courses, the set of all courses is searched for 'same-hour' subsets:

1. Set up the same network as described in the previous section. Arbitrarily pick node i to start. Set j, the same-hour subset to 1.
2. Inactivate all nodes which are directly connected to the distinct node i. Node i is a member of the j-th same-hour subset. Eliminate node i from the set of nodes.
3. Take the next node i not yet inactivated, if there is one, and perform step 2.
4. If there were two or more nodes inactivated start the procedure again: activate all inactivated nodes; j=j+1; arbitrarily pick node i to start go to step 2.
5. If there is only one node left, it is the only member of subset j=j+1;stop. If no inactivated node exists, stop.
6. The minimal number of same-hour subsets is j, given the arbitrary choice of starting nodes.

This lower bound obtained is only an approximation because it depends on the choice of starting nodes. Figure 3.1, subset {120,220,257}, demonstrates this: If course 120 is scheduled at one hour neither 220 nor 257 can be scheduled at the same time. If one starts out by

scheduling course 220 first, course 257 can also be scheduled as there is direct connection between 220 and 257.

Depending on the ordering of the initial set, seven to nine subsets were found using the same data as before. The number of members of each set varied between 80 and 1. This result indicates, that a minimum of seven periods is necessary to schedule the courses using a relaxation of the actual problem with the only constraint that it must be possible for all students to attend all courses of their choice.

# IV. SCHEDULING FINAL EXAMINATIONS

The problem of building a schedule for final examinations is solved here using a heuristic technique. The final examination problem has many fewer constraining factors than the academic course scheduling problem, and therefore infeasibilities or unexpected difficulties are not as likely to occur. Thus, the goal is to design a heuristic which is capable of producing a schedule with as little human intervention as possible.

## A. THE PROBLEM

The final examination scheduling problem may be stated as

**schedule all final exams**

subject to the constraints:

- The finals are written within four consecutive days of one week.
- All courses must have a two hour block of examination time.
- All segments of one course must have the exam during the same period.
- Every student must have enough space to spread out (at least 1.5 places per student).
- There are at most two exams for each student per day.
- The same section must not have two exams back-to-back.
- Faculty members assigned to teach a segment can only be scheduled once per period.
- A course may have more than one room for the exam but all rooms for one segment must be on the same floor of one building.
- On request, particular final exams must be on the first day of the finals week.
- On request, particular courses will have no final exam.
- Rooms should be in designated areas of the campus, according to Table 1.

## B. THE APPROACH

### 1. *Definitions and Notations*

In this section variables are defined and notation is introduced which is used throughout this chapter. A summary is provided in Table 2.

a. Courses

A course is identified by the course number i, i=1,2,...,I, representing the place of the course within the lexicographically ordered set of course names. The number of courses is I. Define the course indicator matrix C to be an I by I 0/1 matrix. Rows and columns of this matrix represent course numbers. $C = \{ c_{ij} \}$ where $c_{ij} = 1$ if course i and course $j \neq i$ have at least one section in common; $c_{ij} = 0$ otherwise. $C_i = \{ c_{i1}, c_{i2}, \ldots, c_{iI} \}$ is the "course indicator" for course i. The number of students in course i is $m_i$.

b. Periods

There are k periods per day, k=1,2,3, where final exams are scheduled. Period 1 and 3 are the two main scheduling periods[1] representing the time period 8AM-10AM and 2PM-4PM. The third period, k=2, is an artifice used to hold courses which would be scheduled at one of the other two periods but a room could not be found or the teaching faculty member is already busy.

By using three periods per day, a total of 12 periods b=1,2,...,12, are available throughout the final week for scheduling. For each of the 12 periods define a "period indicator" $P_b = \{ p_{b1}, p_{b2}, \ldots, p_{bI} \}$ as 0/1 vector such that $p_{bi} = 1$ if course i is scheduled at period b and $p_{bi} = 0$ otherwise.

---

[1]Scheduling two periods during one day ensures that each section has at most two final exams per day.

TABLE 2

NOTATION SUMMARY FOR THE FINAL SCHEDULE

| Subscripts | |
|---|---|
| Course number | i, i=1,2,...,I |
| Faculty member | f, f=1,2,...,676 |
| Room number | r, r=1,2,...,RMAX |
| Day | L, L=1,2,3,4 |
| Period in a day | k, k=1,2,3 |
| All periods | b, b=1,2,...,12 where b=(l-1)3+k |

| Variables | Description |
|---|---|
| $C_i$ | Course indicator for i. $C_i = \{ c_{i1},\ldots,c_{iI} \}$ |
| $P_b$ | Period indicator for b. $P_b = \{ p_{b1},\ldots,p_{bI} \}$ |
| $Y_{fb}$ | Faculty indicator for f at period b |
| $F_i$ | Set of faculty members f teaching i. $F_i = \{ f \}$ |
| $FS_{fb}$ | Faculty time table. $FS_{fb}$ = i if scheduled |
| $x_{rb}$ | Room indicator for room r at period b |
| $S_r$ | Room size of room r |
| $R_i$ | Set of rooms r possible for course i. $R_i = \{ r \}$ |
| $RS_{rb}$ | Room time table. $RS_{rb}$ = i if scheduled |
| $m_i$ | Number of students in course i |

Indicator variable = 1 if scheduled.

c. Faculty

The set of faculty members teaching course i is $F_i = \{ f^{(1)}\ldots f^{(n)} \}$ where $n \in \{2,3,4\}$ consists of one to four different faculty members f out of f=1,2,...,676 members. The faculty indicator $Y_{fb} = 1$ if faculty member f is scheduled at period b; otherwise $Y_{fb} = 0$. If there are n elements in $F_i$, the course consists of n segments requiring at least n rooms for the final. The faculty

schedule $FS_{fb}$ = i if faculty f is scheduled at period b for course i; $FS_{fb}$ = 0 otherwise.

d. Room

The room numbers are indicated by r, r=1,2,...,RMAX. Define the room indicator $x_{rb}$ = 1 if room r is scheduled at period b, otherwise $x_{rb}$ = 0. The size of room r is $S_r$, representing the number of seats. $S_{rmin}$ and $S_{rmax}$ are the lower and upper bound on the number of students room r should hold during finals. The set of rooms situated in the designated area for course i denotes $R_i$. The room schedule $RS_{rb}$ = i if room r is scheduled at time b with course i; $RS_{rb}$ = 0 otherwise.

e. General

A "blocked" course is a course which can not be scheduled because already scheduled courses do not permit scheduling. "Not scheduled" courses have not yet been scheduled. "Unscheduled" courses have already been scheduled but were taken out of the schedule in order to permit the blocked course to enter the schedule.

2. *The Basic Heuristic*

The basic steps of the examination scheduling heuristic are:
INPUT: Data on rooms, faculty and enrollment.
OUTPUT: Room schedule, faculty schedule.

I) Set N = 0 (Number of scheduling tries)
NMAX1 (Maximum number of complete scheduling tries)
NMAX2 (Maximum number of rescheduling tries)
LBLOCK (Limit on number of blocked course for complete retry)

II) Schedule final exams which have to be on the first day.

III) Try to schedule all other courses in listed order (main scheduling heuristic);
N = N + 1.

IV)

1. If (number of blocked courses > LBLOCK) AND (N < NMAX1) then
   reorder the input list such that blocked courses are on top of the list.
   unschedule all courses.
   go to step II).
2. Else, if (number of blocked courses > LBLOCK) AND (N ≥ NMAX1) then
   enter interactive part (the human scheduler decides on the next course of action).
3. Else, if (number of blocked courses < LBLOCK) then,
   N = 0.

V) While N ≤ NMAX2

1. For all blocked courses i
   try to reschedule courses which blocked course i.
   if successful then schedule course i.
   End.
   N = N + 1.
2. If all courses are scheduled then go to step VI).
   Else, if N > NMAX2 then enter interactive part of heuristic.
   Else, if N ≤ NMAX2 then change parameters for scheduling.

End While.

VI) Print schedule and stop.

### 3. *The Main Scheduling Heuristic*

This section describes the main scheduling heuristic which is used to schedule all listed courses. For all courses i, the input list contains:

$C_i$ , the course indicator;

$F_i$ , the set of teaching faculty members;

$m_i$ , number of students in course.

The assignment of suitable rooms, also an integral part of the heuristic, is described separately to enhance the transparency of the heuristic.

**Main Scheduling Heuristic**

(1) $p_{bi}$=0 for all b,i.
(2) For L=1,2,3,4 (all days)
(3) For k=1,3 (period 1 and 3 of a day)
(4) For i=1,2,...,I in listed order (not scheduled courses)
b = (L-1)3+k.
(5) For j=1 to I
if $p_{bj}$ AND $c_{ij}$ = 1 go to step (16).
(6) End (j).
(7) For all faculty $f \in F_i$
(8) if $y_{fb}$ = 1 then begin
b = (L-1)3+2.
(9) For $v \in F_i$ (all faculty)
if $y_{vb}$ = 1 then goto step (16).
(10) End (v).
go to step (13).
(11) End (begin).
(12) End (f).
(13) Assign suitable rooms to course i: Room heuristic.
(14) Update begin
$x_{fb}$ = 1 for all $f \in F_i$
$FS_{fb}$=i for all $f \in F_i$
$p_{bi}$ = 1.
$p_{((L-1)3+k)i}$ = 1.
(15) End (begin).
(16) End (i).
(17) End (k).
(18) End (L).

**Room Assignment Heuristic**

(13.1) Initialize begin
H = number of different elements in $F_i$.

N = $m_i$/H.

r = first element of $R_i$.

(13.2) End (begin);

(13.3) For h = 1 to H, while $r \in R_i$

Call ROOM (N,r).

(13.4) End (h).

(13.5) If yes = 0 then begin

(13.6) For r = 1 to RMAX

(13.7) if $RS_{rb}$ = i then begin

$RS_{rb}$ = 0.

$x_{rb}$ = 0.

(13.8) End (begin).

(13.9) End (r).

go to step (16) (no rooms found).

(13.10) End (begin) (rooms found).

(I.1) **ROOM (n,r) Recursion.**

Set q = r (new first element of $R_i$).

Set yes = 0 (indicates whether room found (1) or not (0)).

(I.2) While $q \in R_i$ AND (q same floor as r)

(I.3) if ($x_{qb}$ = 1) OR ($S_{qmin}$ > n) then go to step (I.9).

(I.4) if ($S_{qmax}$ < n) then begin

n = n - ($S_{rmin}$ + $S_{rmax}$) / 2.

q = q + 1.

(I.5) if (q same floor as q-1) then call ROOM (n,q).

else go to step (I.9).

if (yes = 0) then go to step (I.11).

q = q-1.

(I.6) End (begin).

(I.7) Update begin

$RS_{qb}$ = i.

$x_{qb}$ = 1.

yes = 1 ($S_{qmax}$ < n < $S_{qmin}$)

go to step (I.11).

(I.8) End (begin).

(I.9) q = q+1.

(I.10) End (while).

r = q.

(I.11)End (Recursion).

## 4. *The Decision Heuristic*

The main scheduling heuristic is imbedded within a decision heuristic. Decisions are necessary to determine the course of action to take when an attempt to schedule all courses is unsuccessful.

After an initial attempt to schedule all courses in listed order, the number of blocked courses is counted. The idea is to decide whether to start the process from the beginning with a different ordered input list or to try to schedule blocked courses individually. If there are too many courses blocked, relative to the total number of courses, a complete retry seems reasonable. If the number of blocked courses is above LBLOCK, a limit set for the decision, a retry is performed ((IV.1) basic heuristic):

- Sort input list such that all blocked courses are on top of the ordered input list,
- Initialize all variables,
- Start again with the main scheduling heuristic.

Performing retries in this manner may result in cycling, i.e. after two or three retries blocked courses from the first try may be again among the blocked courses. Therefore, an upper limit on the number of retries, NMAX1, is set. If after NMAX1 tries the number of blocked courses is still above the limit LBLOCK an interactive decision routine is entered ((IV.2) basic heuristic).

If the number of blocked courses during some try falls below LBLOCK ((IV.3) basic heuristic) no complete retry is performed. Rather, the heuristic attempts to individually schedule blocked courses, step (V) basic heuristic:

**Schedule blocked courses individually**

```
For g=1 to G
    For i=1 to I
        For b=1 to 12
            if course i is not yet scheduled then begin
                find number of blocking courses B in period b.
                if B=g then begin
                    schedule all blocking courses in period j, j;b.
                End (begin).
                if all blocking courses are rescheduled then begin
                    schedule i at b.
                    mark i as scheduled.
                End (begin).
            End (begin).
        End (b).
    End (i).
End (g).
```

The procedure is performed with an increasing number of possible courses necessary to reschedule (g) up to a maximum of G which has to be determined by trials. The upper bound G will not be very high for the probability of rescheduling all courses in an already tight schedule decreases rapidly with increasing g. The efficiency of this algorithm can be improved by not only rescheduling course j which blocks course i but also course h which may block course j.

If no complete schedule is found by rescheduling individual courses, a rescheduling of all scheduled courses is performed: An attempt is made to reschedule every already scheduled course i from some period $b_1$ to period $b_2 \neq b_1$. After this change of courses within the existing partial schedule, the individual scheduling of blocked courses is repeated.

If during the scheduling of individual courses a feasible schedule is found the procedure stops; otherwise the interactive

decision part is entered: After displaying course name and number of students of not yet scheduled courses, three alternatives for further action are provided

- **Abort** the scheduling attempt without receiving any results.
- **Retry-and-stop** relaxes $S_{rmin}$ and $S_{rmax}$ for scheduling of blocked courses. It also eliminates the constraint of designated areas while scheduling rooms. It is then tried to schedule blocked courses individually. The scheduling process stops, delivering room and faculty schedules and a list of blocked courses.
- Retry starts the scheduling process again providing choices to set
  * Lower and upper bound on room size $S_{rmin}$ and $S_{rmax}$.
  * Early final exams.
  * The ordering of the input list. This list can be rearranged by random shuffling.

## C. PROGRAM AND IMPLEMENTATION

To decide the question of which programming language to use the handling of data files and the capability to utilize logical operators on bit strings were important criteria. PL1, known for its good file handling capability also provides the variable type BIT, holding strings of up to 256 0/1 variables. PL1 was therefore chosen as programming language; the implementation was performed on a IBM 3033 under the CMS operating system.

Two programs were written, one to supply the course indicator variables and the other to perform the actual scheduling. Input data are supplied by three files; the produced schedules are written into two disk files. A third output file is created in the event of an unsuccessful scheduling attempt. This section describes the data files and the two programs.

### 1. *Data Files*

There are three input files used by the scheduling program, PRIOR DATA, FINAL DTHE and ROOM DAT.. All input files are external stream files. The first row of all files will not be read and the

end of each file will be detected by raising the end of file condition. Descriptions of the files are provided in Tables 3 to 5.

TABLE 3
PRIOR DATA (INPUT)

| Variable | Type | Column | Description |
|---|---|---|---|
| COURSE | Fixed(3) | 1- 3 | Course Number i |
| D(1) Bit(200) | | 6-205 | Bit String Indicator One, $C_i^{(1)}$ |
| D(2) Bit(200) | | 206-405 | Bit String Indicator Two, $C_i^{(2)}$ |
| - | Fixed(3) | 411-413 | Number of Students in i, $m_i$ not read into program |

For the first scheduling attempt, the priority of the courses used to schedule is the same as the sequence of occurrence in the file PRIOR DATA. Data in this file is resorted for further attempts within the program. The user can influence the immediate performance of the program by presorting this input file.

TABLE 4
ROOM DATA (INPUT)

| Variable | Type | Column | Description |
|---|---|---|---|
| ROOMN | Fixed(5) | 1- 5 | Room Number Code r<br>Digit One: Building Number<br>Digit Two-Four: Room Number<br>Digit Five: Room Letter |
| USE Fixed(2) | | 8- 9 | Code for the Use of the Room, i.e. 1 = Lecture Room |
| PLACES | Fixed(2) | 17-18 | Maximum Number of Places $S_r$ |

TABLE 5

FINAL DTHE (INPUT)

| Variable | | Type | Column | Description |
|---|---|---|---|---|
| COURSE | | Fixed(3) | 1- 3 | Course Number i |
| CN | Char(6) | 9-14 | | Course Name |
| N | Fixed(3) | 20-22 | | Number of Students in i, $m_i$ |
| COND | Fixed(1) | 35 | | Condition, Indicating Team Teaching(1), or Lab (2) |
| FA(1) | CHAR(2) | 40-41 | | First Faculty Member, $f^{(1)} \in F_i$ |
| FA(2) | CHAR(2) | 45-46 | | Second Faculty Member, $f^{(2)} \in F_i$ |
| FA(3) | CHAR(2) | 50-51 | | Third Faculty Member, $f^{(3)} \in F_i$ |
| FA(4) | CHAR(2) | 55-56 | | Fourth Faculty Member, $f^{(4)} \in F_i$ |
| FI | Fixed(1) | 60 | | Final Written (0), not Written (1) |

As output files a room file, DRUCKR DATA and a faculty file, DRUCKP DATA were chosen. They provide the necessary information for students and faculty members. The implicit assumption is that the faculty members teaching different segments of one course decide among themselves which rooms of the designated ones will be used by the students of their respective segments.

The key of the faculty file is the mailcode of the faculty member which indicates the row. Columns are the different final examination periods. If a final is scheduled for a faculty member at one period, the course name is entered at the name/period intersection.

For the room file, the key is the room number indicating the row. The columns are also the final periods. Course names are entered where courses are scheduled. Examples of the output files DRUCKR DATA and DRUCKP DATA are provided in Appendixes B and C.

## 2. *Program to Prepare the Input File*

The program PRIOR PLC prepares the input file PRIOR DATA for use in the main program FINAL PLC. As input, an ordered list is used containing the course name followed by the segment number and number of students in the segment. The output file is the file PRIOR DATA described in Table 3.

Up to 6000 entry sets, course name, section name and number of students in section stored in the structure MATRIX (*) are read from the disk. Courses which need not to be scheduled are sorted out. Then, the total number of students in each course TOTAL(K) is calculated. The array CN(K) is filled with the course name, corresponding to the course number K. TOTAL, CN and K are the entries in the output file CROSSL DATA, which is for software maintenance purposes only.

The array B1(j), j=1 to 200, where the only nonzero bit of 200 is at the $j_{th}$ place. These entries are used to create the array D(*,*), the course indicator.

The program runs under batch. The input file is included in the job stream of the program by the time of submission. The program and a small sample of the input file is attached as Appendix D.

## 3. *Program to Schedule the Final Examinations*

The PL1 program FINAL PLC, provided as Appendix A, consists of a main routine and 19 internal subroutines sorted and described in alphabetical order. The program is written in the PL1 subset PLC, runs under VM/CMS and does not use any external or library subroutines. The three input files are read completely into memory. File definitions (FILEDEF) are provided in an EXEC file which is part of the introductory comment of the program. The following sections describe the programs with respect to special features only. Bounds and other parameters were found empirically during trial runs.

a. The Main Routine

During the reading of the input file ROOM DATA, only rooms suitable for lecture type-teaching are stored into arrays. Rooms which can not be used during final examination week are filtered out. A way to indicate this externally is to set the USE indicator in the ROOM DATA file to a number greater than ten. The files PRIOR DATA and FINAL DATA are read without further manipulation.

The first task in the main routine is to create the array B1. It is an array of 200 bit strings filled with 199 zeros and one '1' where B1 (1) has the '1' at the most significant bit and B1 (200) the '1' at the least significant bit. The elements of this array are used to mark scheduled courses.

After the label TRY, courses which do not have a scheduled final (F1=1) or which are unscheduled (FA(*,1)=0) are eliminated: Each bit indicator D(*,*) is set to '0' at the position of these courses. The course number of these courses is set negative to indicate no further consideration.

The variables DOWN, V, V1 and V2 are used to control the program. DOWN is set to '3' when all the prefered building constraints are to be honored, and'0' otherwise. V indicates the number of tries to schedule all the courses, V1 and V2 count subtries within each V.

When first entering the main routine, the subroutine EIN is called, which is used to interactively read courses from the screen which should be scheduled during the first day of final examinations. Then, the subroutine PARTIAL tries to schedule all courses and COMPLET checks whether or not all courses are scheduled. If all courses are scheduled, the output files are written and the program terminates.

In the case, that there are some courses not scheduled, further computation depends on the number RE of not scheduled courses. If the number is above 5% of the number of total courses, the bounds on the room size (LOW,HIGH) are relaxed and the scheduling of all courses is done again. The difference to the scheduling run before

is that after scheduling early finals, blocked courses are scheduled first. This sequence is performed up to 5 times (counted by V1). If the number of not scheduled courses is still about 5% the interactive part of the main routine is entered.

If the number of not scheduled courses falls below the 5% level subroutine NEIGH is called. NEIGH tries to rearrange already scheduled courses in order to make it possible to schedule the remaining courses. This is done up to five times (counted by V2) where the number of courses being rescheduled grows from try to try. After every try, COMPLET is called to check whether all courses are scheduled.

After trying to schedule all courses with NEIGH unsuccessfully, a last attempt is made by calling AGAIN. There, all already scheduled courses are rescheduled one by one, if possible, in a 'wholesale' fashion. After that, NEIGH is called once more. If there are still some courses blocked the interactive part INTER is entered.

In the interactive part, all not scheduled courses are written to the screen along with the number of students in the course. For scheduling purposes one can decide between three alternatives: "Try again", "stop and print" and "stop". Stop just terminates the program. Print and Stop writes the incomplete schedules along with a file of not scheduled courses (FAIL DATA) to the disk and terminates after having called NEIGH with further relaxed bounds on the room size.

The option 'Try again' opens more choices. One can reset the early finals and the input file PRIOR can be sorted 'semi-randomly' by a call to SWITCH. Depending on the choice, initial conditions are reconstructed and the scheduling process starts again at the label TRY.

b. The Subroutine AGAIN

A wholesale rescheduling of already scheduled courses is performed. For every course with no early final, an attempt is made to find a schedule in another period. If it is possible, the old entries in the room array TIMER and faculty member array PTIME are canceled.

The scheduling is done by the subroutine GUIDE, the canceling of TIMER is performed in this subroutine and the canceling of PTIME in subroutine LOESCHP.

c. The Subroutine COMPLET

The subroutine goes through the array COURSE (*) and notes all nonnegative course numbers, corresponding to not scheduled courses. The counter RE counts the number of not scheduled courses, the flag FLAGCCC indicates when all courses are scheduled and the array REST(*) holds the course numbers of the not scheduled courses for further use.

d. The Subroutine EIN

This subroutine enables interactive entry of courses which have to be scheduled early. These courses are stored in the array EARL. The subroutine asks for the course name of the course in question, tries to schedule it during the three periods of the first day and writes the result of the attempted to the screen. The subroutine provides for 20 courses, which is a upper limit due to past experience. It could easily be extended by changing the outer DO-Loop. The variable I5 holds the actual number of early finals.

e. The Subroutine GUIDE

This subroutine regulates the scheduling of one course during one period. The hour indicator HB(Period,1) and HB(Period,2), each a bit string of 200 bits, have '1' at the place of the course number already scheduled at this period. To find out whether the course in question can be scheduled at a particular period the course indicator D(Course,1)[2] and D(Course,2) are logically ORed with HB (*,*). If the result is false, there is no conflict with other courses in this period. To find out whether the teaching faculty members are idle

[2]The course indicator for each course had to be broken up into two pieces D(200,2) because of the upper limit on bit strings in PLC.

and whether rooms can be found during this period the subroutine PROFSUB is called.

If the courses conflict, a faculty member is not idle or rooms cannot be found during this period and the in-between period (k=2), the program returns to the calling routine. Otherwise the hour indicator HB(*,*) is updated: the B1(*) entry corresponding to the course number is logically ORed with HB(*,*). The course number is set negative to indicate that the course is already scheduled.

f. The Subroutine DRUCK

The subroutine is called to write the room and faculty schedule into the external files DRUCKR DATA and DRUCKP DATA.

g. The Subroutine LENK1

This subroutine is called on the second and every further complete try (V>1). It schedules at the beginning of each try the early finals and not scheduled courses of the previous try. The I5 early finals stored in array EARL are scheduled in the three periods of the first day while the RE previously not scheduled courses stored in REST are tried to be scheduled during all available periods.

h. The Subroutines LOESCH and LOESCHP

Both subroutines are used to erase already scheduled courses out of the arrays TIMER {LOESCH} and PTIME {LOESCHP} during a given period. All entries of the respective array are searched during this period for occurrences of the course. The entries are set to zero.

i. The Subroutine NEIGH and NEIGH1

The subroutine NEIGH performs the task to schedule a not scheduled course, REST(R3), during a period R1 by first rescheduling courses which conflict with course REST(R3) and then schedule the it. Depending on V2, one to five (RR) courses are tried to reschedule:

First, conflicting courses during this period are found and stored into the array R4(*). If the number is greater than the upper bound V2 during this try the next period is tested. If there are no course conflicts (R4(1)=0) a scheduling attempt is made. If RR>0, the program checks whether all the teaching faculty members for the course in question are idle. In case all faculty members are free, a rescheduling of the conflicting courses is attempted during another period.

For every conflicting course all periods are tested. In a recursive fashion the subroutine NEIGH1 is called which performs the same task as NEIGH for courses to be rescheduled.

If a course is rescheduled, the old entries are erased in TIMER and PTIME by LOESCH and LOESCH1 and the hour indicator HB(*,*) is updated. If all conflicting courses are rescheduled, the not scheduled course in question is scheduled and all arrays are updated. The whole procedure is done for all entries of the array REST.

j. The Subroutine PARTIAL

PARTIAL provides the loops to schedule all courses. It calls GUIDE for each period and each not scheduled course.

k. The Subroutine PROFSUB

The mailcodes of the at most four teaching faculty members of a course are transformed into a corresponding integer. For each faculty member, it is checked whether he/she is idle during this period by checking the entry of PTIME.

If all faculty members are idle, the number of segments is found: More than one segment is only assumed if there is more than one different faculty member and the condition COND does not indicate lab or team teaching (EQ>1).

For each segment an equal share of students is calculated (NQ) and for each segment the subroutine ROOMSUB is called. If all segments can be scheduled during the same period, entries in the array PTIME are made. Otherwise entries in the array TIMER, made in one

of the ROOM subroutines, are erased by calling LOESCH and the program returns.

l. The Subroutine SWITCH

This subroutine uses the PLC function RAND(X) to resort the arrays COURSE(*) and D(*,*) pseudorandomly. Arrays M1(*) and M2(*,*) are created to hold the above arrays temporarily.

m. The Subroutine ROOMSUB

Depending on DOWN this subroutine sets the bounds on designated areas for rooms first and then calls ROOMS1 if DOWN=1, or calls ROOMS1 immediately. For each type of course, the building bounds (ROOMP,ROOME) are stored. The decision to use one of the sets is based upon the first two letters of the course name. A priority among different possible buildings or floors is reached by calling the subroutine ROOMS1 first with the set of bounds for the highest priority. If enough rooms are not found, ROOMS1 is called again with a new set of bounds, lower in priority.

n. The Subroutines ROOMS1 to ROOMS4

These four subroutines perform essentially the same task. Looking at each room between the upper and lower building bounds (ROOMP,ROOME), they search for an empty room not too big (≤HIGH). If a room with these specifications is found it is checked whether it is too small (<LOW). If it is not to small, the room is scheduled by entry into the array TIMER and the program returns;otherwise the next higher version number of ROOMS* is called to look for an additional room on the same floor holding the rest of the segment. This can be done for up to four rooms.

To avoid unnecessary searching for very small course sizes the variable SINGEL declares a fixed upper bound on the room size for which a room is declared too big. Thus, a course of four students will be scheduled in a room with up to 26 places if SINGEL is set to 26. If a room is to big and its size is above SINGEL, it is not

considered for this course. The routine looks at the next course. Upper and lower bounds (LOW,HIGH) as a factor on course size can be set and are changed during the execution of the program.

o. A Maintenance Toggle

For maintenance purposes a toggle is built into the program to make a run somewhat more transparent and provide for more choices. By setting the variable PRFLAG=1 printouts of intermediate steps is displayed on the screen. After every try, all not scheduled courses are displayed, together with information on which of the subroutines was trying to schedule this particular, not yet scheduled course. When entering the interactive part answering the question 'TRY AGAIN????' with "2" branches into a selective scheduling option providing even more possibilities for the skilled user.

## 4. *Computational Results*

All programs and subroutines where tested using real data from quarter II/85. A comparison to the manually built schedule of the same quarter however was not performed because no measure of effectiveness other than feasibility exists.

The preparation of the input data was performed using the PLC program PRIOR. The input to this program came from the existing manual system: a file of punch cards called "Y-Cards" where each card holds the course name, one section enrolled in this course and the number of students in this section. The program PRIOR sorts out courses which are known to have no final exams that quarter and builds the two output files PRIOR DATA and CROSSL DATA, described in Chapter IV.C.1. The number of Y-Cards for the test data was 3843 with 267 courses and 850 sections remaining for scheduling of final examinations. The run time for this data set on the IBM 370 batch processor was 12 minutes.

To create the FINAL DTHE file, data from the questionnaire file can be used as soon as this system is installed and information is updated. For test purposes the file CROSSL DATA was manually edited

and brought into the format of the FINAL DTHE file (Table 5). The ROOM DATA file was created by using inputs from the scheduler's file cards; 61 rooms were available for final examinations.

The run time of the final examination scheduling program FINAL PLC varied between 90 and 120 seconds, including read/write operations. The variability is due to the subroutines used by the program control. The performance depends on the initial ordering of the input file PRIOR DATA. If a run cannot produce a feasible schedule, reordering of this file ( by course size, number of sections in course or just randomly) tends to give an immediate improved result. A schedule with a few courses not scheduled (output to FAIL DATA) can usually be converted into a feasible schedule by restarting the program and scheduling some of the courses from FAIL DATA as early finals. The interactive part of the program provided adequate options for the test data to reach a feasible schedule independent of the initial ordering of the input data. For the test problem a feasible schedule was found; parts of the resulting schedules are attached as Appendixes B and C.

# V. SCHEDULING ACADEMIC COURSES

The main scheduling task every quarter is concerned with the academic courses. Currently, this is done manually and takes about four weeks, the room scheduling included. Additional time is needed to prepare the file cards and for the writeup of the master schedule. During the scheduling period, the registrar process is stopped with respect to updates of sections. This leads to schedules which are not up-to-date, contributing to the high amount of changes experienced at the beginning of each quarter.

To decrease the time for building a schedule, an approach is described which would computerize the whole scheduling process. Results reported in the open literature, previous experience with the computerization of the scheduling process at the NPS (Chapter I.C) and the analysis of the course-section interaction (Chapter III) lead to a heuristic approach.

Due to the short time available and the scope of the problem, it was not possible to fully design an algorithm or to implement it on a computer. Instead, an approach is described and a data structure is proposed although several subproblems are solved and implemented:

- -The collection of the data, described in Chapter II.C.
- -The search for suitable rooms and the assignment of rooms to segments; with minor changes like implemented in the final examination program.
- -An integer linear programming approach to the single course scheduling problem; formulated and solved as an example within this chapter.

## A. THE PROBLEM

By combining standard settings, frequent constraints (Chapter II.B.2) and logical constraints scheduling of academic courses can be completely described as follows:

**Schedule all required courses**

subject to the constraints:

- All sections are scheduled such that every course of the approved sequence can be attended.
- Rooms can only be assigned to one segment at the same time.
- Rooms must be large enough to hold all students and suitable for the course purpose.
- Rooms must be in designated areas of the campus (Table 1).
- Segments of a course must be of approximately equal size. On request, segments consist of specified sections.
- Faculty members, assigned to teach a segment, can only be scheduled once per period.
- Every course must be scheduled with as many lecture and/or lab hours per week as indicated in the current catalog of the NPS or requested by the teaching faculty member.
- Regular meetings, seminars and guest lectures must be incorporated into the schedule.
- Faculty and students must be allowed one lunch hour in the period between 11AM and 2PM.
- There are nine one-hour periods per day, five days per week. The schedule repeats each week.
- A teaching block for a particular segment consists of one to four consecutive hours, independent for lecture and lab part. Each block is scheduled at most once per day.
- Segments of courses with lab parts must be subdivided in equally sized subsegments if requested.
- Accelerated and refresher courses may be scheduled pairwise at the same time and into the same room.
- Courses are scheduled together at the same time and location on request.
- Teaching and nonteaching periods of faculty members may be requested by hour and day.
- Faculty members participate in courses as student.

- Segments are scheduled with two faculty members teaching in a team on request.
- Different segments taught by the same faculty member are scheduled back-to-back or with a gap on request.

## B. THE APPROACH

The driving force of this heuristic approach to the problem is the necessity to produce a schedule which is feasible and includes all courses. To reach this goal, interaction between the human scheduler and the scheduling program must be possible at every step of the process to incorporate all constraining factors and eventualities occurring during the scheduling process.

If infeasibilities within the data set should arise during the scheduling, it must also be possible to contact the source of the data and reiterate the scheduling process with changed inputs. As this takes some time, the scheduling process will last for several days, despite the use of a computer. Hence, the issue of an easily maintainable and accessable data base is of major concern, too.

### 1. *The Basic Heuristic*

Figure 5.1 shows the flow diagram of the proposed scheduling system.The heuristic is based on the serial scheduling of complete courses: A course is scheduled with all its segments at the same time. While scheduling a particular course, only constraints of this course and conflicts with already scheduled courses are of concern. This has the disadvantage of shortsightedly creating problems to courses yet to be scheduled. This disadvantage could only be completely avoided by the use of mathematical programming techniques, which were ruled out earlier. Some measures taken, like priority scheduling, try to find a way around this problem. After a short description of the heuristic, each block of it is discussed in more detail in the following sections.
The first step of the heuristic is the processing of the rough data (A). All necessary data concerning courses,faculty,sections,rooms and constraints are checked for completeness and consistency. The data is

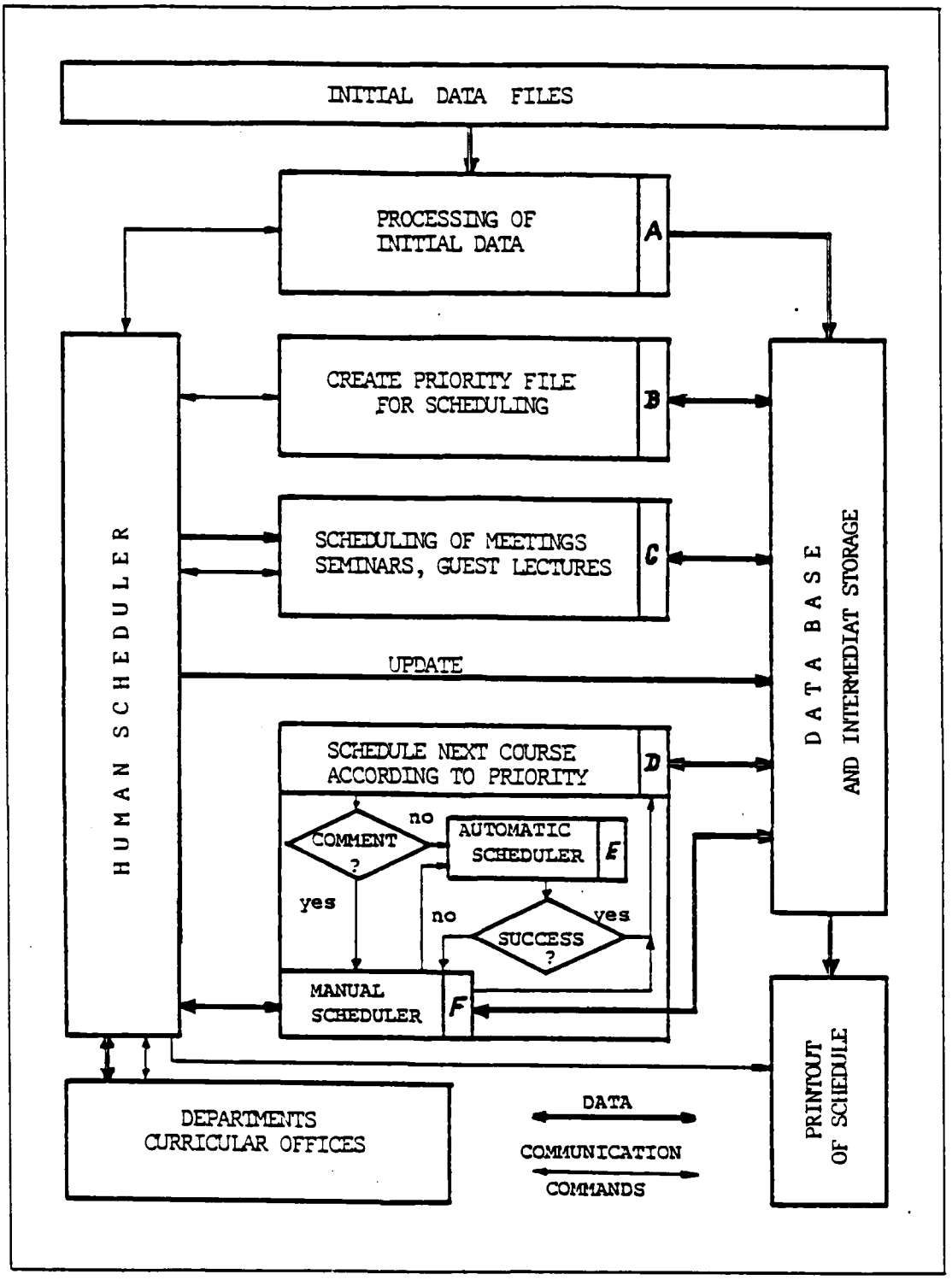



**Figure 5.1 Flow Chart of the Scheduling Heuristic.**

rearranged into different files of the data base used during all further steps to ensure minimum core requirement, fast access and easy update. The data base is stored and completly reloaded when breaks occur during the scheduling process.

The control of the scheduling is performed by a priority system (B). A priory list is built to decrease the amount of friction experienced throughout the process. Valuable experience gathered by the schedulers over the last decade is used to order the courses in this file. The file is also used to ensure continuity after breaks within the process: Already scheduled courses are marked and the last situation is stored.

Regular meetings, seminars and guest lectures (C) are scheduled before the course scheduling is started. The reason for this high priority is the envolvement of big groups within the faculty or even the whole student body. For almost all meetings, seminars and guest lectures, time, place and audience is fixed, leaving no choice in the scheduling process. Therefore an interactive, menu driven scheduling routine is appropriate.

The scheduling of each individual course (D-F) involves several steps and must be highly interactive. First, the constraint set has to be checked for comment-type constraints. If there is a comment, a automatic scheduling can not be started until it is "translated" into data format. The proposed method is a menu driven, interactive heuristic to translate the comments and schedule the course (F). To decrease the amount of necessary data displayed to the scheduler on a terminal, all courses with comment-type constraints receive highest priority. This reduces the amount of interaction with already scheduled courses and thus the amount of essential data needed to schedule the course.

A course with no comment-type constraints is scheduled by an automatic routine (E). If no feasible solution can be found, the control is handed back to the manual scheduler (F). Now, all information has to be displayed necessary to enable the scheduler to detect the infeasibility and resolve the problem. This may involve unscheduling of

already scheduled courses, resorting of sections within segments of already scheduled courses, relaxing of constraints, or even calling the respective curricular offices of the students or departments of the faculty members involved to reach alternate decisions. The diversity of possible solutions to the problem of scheduling "difficult courses" forces an interactive approach.

After reaching a solution and also during the process, the printout and storage of results must be accomplished easily. Changing parts of a completed schedule must also be possible as well as an update of the data base. This can be accomplished by an editing routine.

### 2. *Data Base*

The data base has to hold all the necessary data, must be loadable entirely into memory and must be easily maintainable and accessable. The proposed data base accomplishing these requirements is shown in Figure 5.2 and described in this section. The data base is split into several files, connected by pointers. The files are:

- COURSE
- SEGMENT
- SECTION
- CONSTRAINT
- COMMENT
- FACULTY
- ROOM
- PRIORITY

The purpose of splitting the data base into different files is the saving of space. By not allocating space for every possible segment per course, comments per faculty member, etc., the amount of space used by the data base is proportional to the actual amount of data, not to the possible amount.

For purposes of this description, define a "record" to be a set of data of the same kind, belonging together. A record is recognized by a key or the place in a file, which holds records of the same structure.

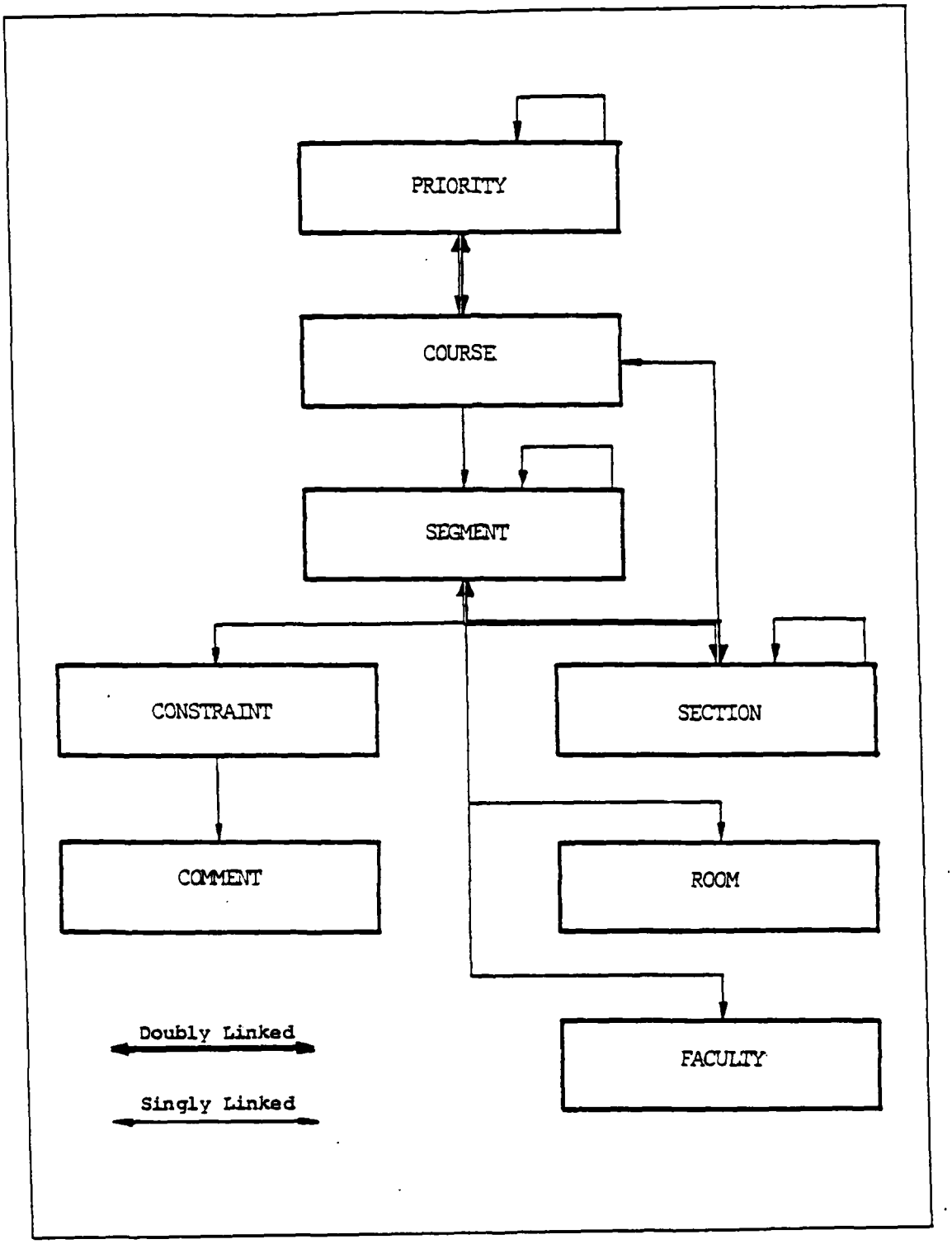



**Figure 5.2 Representation of the Data Base.**

a. Course File

The key for the course file is the course number, COUNO, a three digit number corresponding to the course name. Under this key, records in Table 6 are defined.

TABLE 6

DATA BASE, FILE COURSE

| Name | Description |
|---|---|
| NOSEG | Number of segments in course |
| NOLAB | Number of lab subsegments per segment |
| HOULEC | Number of lecture hours per week |
| HOURLAB | Number of lab hours per week |
| NOCOU | Number of students in course |
| COUSEGP | Points to first segment of this course in segment file |
| Key: COUNO | Course number; file sorted by COUNO |

COUNO need not be stored explicitly if the course numbers are continuous integers, starting at a predefined value and the file is sorted in increasing order with respect to the course number. This implies, that record number k coincides with course number k. All segments of a course can be found by offseting the pointer to the segment file, COUSEGP, by 0 to NOSEG-1, because segments of a course are stored consecutivly following the segment pointed to. About 400 records need to be stored.

b. Segment File

The segment file consists of one record for each segment. Records can be identified by a pointer from the course file. Segments of one course are stored consecutively. A record is described in Table 7. For the lecture part of the segment, the record provides for possible team-teaching of a segment by defining fields for faculty member and constraint pointer twice. Each faculty member is identified by a unique three digit number corresponding to his/her two letter

mailcode. If only one faculty member is teaching, PRLEC2 is marked. CON* points to the respective record of the faculty member in the constraint file. For the lab part of a course there is usually no team-teaching and thus, only one faculty identifier PRLAB and pointer CON3 is defined. For lecture and lab part there is one pointer each to the assigned room record in the room file. SEGSECP points to the first section of the section file belonging to this segment. About 500 records are necessary.

TABLE 7

DATA BASE, FILE SEGMENT

| Name | Description |
|---|---|
| PRLEC1 | First faculty member teaching lecture part |
| CONP1 | Points to Constraint file, constraints of PRLEC1 |
| PRLEC2 | Second faculty member teaching lecture part |
| CONP2 | Points to Constraint file, constraints of PRLEC2 |
| ROOMLECP | Points to Room file, assigned lecture room |
| PRLAB | Faculty member teaching lab part |
| CONP3 | Points to Constraint file, constraints of PRLAB |
| ROOMLABP | Points to Room file, assigned lab room |
| SEGSECP | Points to Section file, first section of segment |

Key: none, pointed to ; sorted by Course by Segment

c. Section File

The section file holds individual records for about 1000 sections. Sections are identified by the place in the section file, which is ordered by section number. Each record provides for a maximum of six courses which can be taken by any student. COUSEGNO* is a combination of the course number COUNO (first three digits) and the running number of the segment (last digit). A pointer associated with each COUSEGNO points to the next section of the same course/segment. NOSEC holds the number of students in the section. F(weekday,hour) holds an indicator for every hour of the week. The indicator shows the state of the scheduling process (Table 9) and the whole field F(*,*)

will hold the time table for each section at the end of the process. Table 8 describes a record.

TABLE 8

DATA BASE, FILE SECTION

| Name | Description |
|---|---|
| COUSEGNO1 | Course and segment number of first course. |
| SECP1 | Points to next section of same course/segment. |
| . | |
| . | |
| . | |
| COUSEGNO6 | Course and segment number of sixth course. |
| SECP6 | Points to next section of same course/segment. |
| NOSEC | Number of students in section. |
| F(1,9) | Timetable of Monday. For each hour an indicator shows, which course is scheduled. |
| . | |
| . | |
| . | |
| F(5,9) | Timetable of Friday. |

Key: section number file sorted by section number.

TABLE 9

USED INDICATORS IN FIELD F(*,*) OF FILE SECTION

| Indic | Description |
|---|---|
| 0... | open for scheduling |
| 1... | COUSEGNO1 scheduled |
| . | |
| . | |
| . | |
| 6... | COUSEGNO6 scheduled |
| 7... | lunch scheduled |
| 8... | meeting scheduled |
| 9... | guest lecture scheduled |

d. Constraint File

This file has no special order for it will only be accessed via pointers from the segment file. For each segment and teaching faculty member one record (Table 10) is necessary, holding the constraints imposed by the faculty members. As the constraint file is constructed out of data collected by the proposed system described in Chapter II.C.2, detailed information about each field can be found in Appendix G. About 500 records are necessary.

TABLE 10

DATA BASE, FILE CONSTRAINT

| Name | Description |
|---|---|
| SPECSTUD | Indicates whether a particular group of sections needs to be in a segment. |
| TEAM | Indicates whether course is taught in a team. |
| SPDAY | Indicates preferance of a day. |
| SPHOUR | Indicates preferance of a hours. |
| BACKTOBACK | Indicates whether back-to-back teaching with other segment of same faculty member is preferred. |
| ACCEL | Indicates accelerated or refresher course. |
| TOGCOU | Course number of course to be scheduled with this course. |
| BLOCKLEC | Block length of lecture part in hours. |
| BLOCKLAB | Block length of lab part in hours. |
| RLEC | Room needed for lecture (first choice). |
| RLEC | Room needed for lecture (second choice). |
| RLAB | Room needed for lab (first choice). |
| RLAB | Room needed for lab (second choice). |
| CONCOMP | Points to comment, if one exists. |
| Key: none | unsorted; pointed to from segment file. |

e. Comment File

The comment file holds comment-type constraints which could not be easily discretized. Each record is pointed to by CONCOMP from the constraint file. About 50 records are necessary.

f. Faculty File

This file is sorted according to the faculty number corresponding to the faculty mailcode. Each record holds the faculty member's department number, DPNO, and a compressed timetable, P(weekday,hour), where entries are:

$$P(\text{weekday,hour}) = \begin{cases} 1 & \text{if already scheduled for all weekdays and hours} \\ 0 & \text{if not yet scheduled} \end{cases}$$

As all combinations of the two-letter mailcode are, used for the faculty number, 676 records are necessary.

g. Room File

The key of the room file is a five digit room number. The file is sorted according to this room number. A record is described in Table 11. There are 120 rooms used for academic courses.

TABLE 11

DATA BASE, FILE ROOM

| Name | Description |
|---|---|
| ROOMNO | Room number |
| NOPLACES | Number of places |
| KIND | Kind of room (lecture,lab,etc.) |

$$R(\text{weekday,hour}) = \begin{cases} 1 & \text{if already scheduled for all weekdays and hours} \\ 0 & \text{if not yet scheduled} \end{cases}$$

Key: ROOMNO room number ; sorted by ROOMNO

h. Priority File

The key for a record is the course number, COUNO. The file is sorted according to this number. A record contains the course name, COUNAME, and a pointer to the next record, PRIP, with lower priority. The zeroth record, with a dummy entry in COUNAME, points with its entry in PRIP to the highest priority course.

i. Creating the Data Base

The task is to set up the data structure and fill it with selected data from registrar files and the constraint set as well as to connect the files by pointers. Although this is not particularly difficult, it is crucial at this point to check all data for easily detectable inconsistencies and infeasibilities and to explore ways to incorporate some special features of the data to ease further scheduling. Some points to check and consider are:

- Check, whether faculty member teaching a course in a team have enough overlapping time to teach.
- Check, whether rooms which are specially requested are large enough and whether all courses can be taught in this room within the 45-hour week.
- check, whether all necessary data are supplied.
- Lecture and lab hours are supplied by the current NPS catalog. Some faculty member need more or fewer hours. This supercedes the catalog data and should be used in their place.
- Faculty member may participate in courses as student. A possible implementation of this constraint is to add an additional record to the section file representing this faculty member and the course he/she wants to take. A connection between this record and the record of this faculty member in the faculty file must be made. For this purpose one of the six field pairs (COUSEGNO*,SEGP*) can be used as they will not be needed entirely. This will ensure automatic consideration of these constraints without creating an entirely different case within the heuristic.

### 3. *Assigning Priorities*

In this sequential heuristic approach, the quality of the solution and the speed in which it is reached will depend highly on the order the courses are processed. The proposed heuristic uses a two step priority to determine the sequential ordering:

- a static priority file and
- a "time dependent" secondary step.

Both steps attempt to emulate the current manual scheduling process. The currently used priority setting is included as Appendix I. After implementation of the scheduling system, experience will show whether this concept has to be changed entirely or how it must be modified to reach satisfactory results.

#### a. Static Priorities

The set of courses is sorted according to priority criteria such that courses with a tendency to create difficulties during scheduling will be scheduled first. The ranked criteria are as follows:

1. Courses with comment constraints.
2. Refresher courses and accelerated courses.
3. Courses with lab parts. Within this group, labs with blocks of three hours rank higher than those with blocks of two hours.
4. Courses which require a certain day of the week
5. Courses taught in a team.
6. Courses with lecture blocks of two hours or more.
7. All other courses.

A tie within a priority group is broken in such a way that courses with a higher student-to-segment ratio will be assigned a higher priority.

#### b. Time Dependent Priorities

While building the static priority file, no criterion is used to incorporate the element of time into the ordering. This is done at run time.

Starting at 8AM, the first period, all courses are scanned in order of the priority file, to find a course which must be scheduled

at this time. These courses are scheduled first. A second scan for the same period looks for courses which have constraints prefering this time. During a third scan courses which are not yet scheduled but do not have a preferance for all coming periods are considered for scheduling. This procedure is repeated for each period.

### 4. *Scheduling Seminars, Meetings and Guest Lectures*

There are regularly held seminars, meetings and guest lectures concerning different subsets of the faculty and the student body. For most of them, everything is predetermined, leaving little choice for the scheduler. The lack of freedom in scheduling and the exceptionally large groups involved makes it necessary to schedule them first. The proposed solution to this part is a menu driven, interactive routine sketched in Figure 5.3.

Step (2) of the interactive routine asks for the weekday and the hours of the event. The kind of event is asked for in step (3) and the group involved in step (4). If the faculty of a whole department is involved steps (5) and (6) apply: Field P(weekday,hour) for all members, depending on the department number, DPNO, is marked with a '1'. Particular faculty members, steps (5), (8) and (9), are found by their mailcode equivalent, determining the place within the ordered faculty file. Here, too, P(*,*) is set to '1' at the indicated day and hours.

As soon as all students are involved in an event, steps (5) and (10), F(weekday,hour) of all students is marked with a '9' in the appropriate field indicating a guest lecture. Seminars, steps (5), (11) and (12) have course numbers and course names just like regular academic courses. This course number has a record in the course file and thus all sections are reachable via pointers. By supplying the course number, seminars are scheduled like regular courses, without involvement of particular faculty members.

After scheduling of one event, the menu repeats itself until all meetings, seminars and guest lectures are scheduled.

```
(1)     Start

(2)     When:   Day: (?)  Hour1: (?)  Hour2: (?)  Hour3: (?)
(3)     What:   Seminar (8), Meeting (1), Guest Lecture (9): (?)
(4)     Who:    Student (5), Faculty (11): (?)

on last answer goto step (5) or (11)

(5)     Who:    Whole Department (6) or Particular (8): (?)

on answer got step (6) or (8)

(6)             Department number: (??)

assign to all faculty members of department '1'
in F(weekday,hour) for the given day and hours.

(7)             Another Department for same day/hour Y/N: (?)

if Y goto step (6)
if N goto step (13)

(8)             Mailcode: (??)

assign to particular faculty member '1'
in P(weekday,hour) for the given day and hours.

(9)     Another Person for same day/hour Y/N: (?)

if Y goto step (8)
if N goto step (13)

(10)    Who:    All ( ) or Particular ( ): (?)

if ALL then
        set F(weekday,hour) for all students
                to '9' at given day and hours;
        goto step (13).

(11)            Which Course: (??????)

set P(weekday,hour) for all sections taking this
course at given day and hour;
mark course in priority file as scheduled.

(12)            Another course, same time Y/N: (?)

if Y goto step (11)
if N goto step (13)

(13)            Everything scheduled? Y/N: (?)

if Y goto (14)
if N goto (2)

(14)    Stop.

                Bold face indicates questions on screen.
```

Figure 5.3 Seminar, Meeting and Guest Lecture Scheduling.

## 5. *The Automatic Scheduling of a Course*

The scheduling of any course involves three major steps:

- distributing sections among the segment of a course,
- finding suitable periods for sections and faculty,
- assigning rooms.

These problems have to be solved subject to the constraints defined in the first section of this chapter. The proposed system provides for an automatic and manual system. Except for courses with comment-type constraints, the scheduling of all other courses is first tried automatically. Only in the case of failure is the manual scheduling routine entered.

The distribution of sections among segments is closely connected with the problem of finding a suitable period because only sections with a common idle block can be sorted into one segment. As a solution technique integer linear programming, ILP, can be employed, where the stated problems and constraints are incorporated:

- A course may have between one and four segments.
- The sections of a course must be distributed roughly evenly among the segments with respect to the number of students.
- Faculty members assigned to teach the course must be assigned to a particular segment.
- Sections and faculty members can only be assigned to periods where both are still idle.
- Different block sizes have to be observed as well as possible back-to-back constraints.

Due to the different block sizes and varying number of faculty members and segments, a single ILP formulation is not possible. With respect to the number of faculty members and number of segments experience shows that the listed cases are possible:

| Case | Faculty | Segments |
|---|---|---|
| (I) | 1 | 1 |
| (II) | 1 | 2 |
| (III) | 2 | 2 |
| (IV) | 2 | 3 |
| (V) | 3 | 3 |

| | | |
|---|---|---|
| (VI) | 2 | 4 |
| (VII) | 3 | 4 |
| (VIII) | 4 | 4 |

The block size leads to further increase in number of cases: blocks of 1,2 and 3 consecutive hours are possible. Figure 5.5 shows the ILP formulation for the sample case VI. Within the next sections special considerations concerning the different cases and some other constraints in connections with the ILP are discussed.

a. Periods and Blocks

In general, 45 one hour periods are available during the scheduling horizon of one week. However, as policy it is tried to schedule courses during the same period every day, i.e. the four single hour blocks of course XN3020 from Monday through Thursday between 9AM and 10AM. Therefore, the ILP is based on one "scheduling day," the sum of the scheduling days of the week. Figure 5.4 shows an example: For a course with three one-hour blocks, Wednesday,Thursday and Friday, already scheduled courses are added into the scheduling day (block size 1), leaving periods 2,6,7,8,9 open for scheduling.

As very few courses consist of five one hour blocks, the addition of all already scheduled periods of the week into the scheduling day would unnecessarily decrease the freedom to choose in the ILP. For the different number of blocks per week a course can have, the following list provides the number of possible combinations of week days summed into the scheduling day:

| Number of blocks: | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Number of combinations: | 5 | 10 | 10 | 5 | 1 |

If the rule is employed that consecutive days early in the week are preferred, the following algorithm can be introduced:

1. The first scheduling day is the sum of the first n consecutive days of the week (n = number of blocks during the week).
2. Solve the ILP

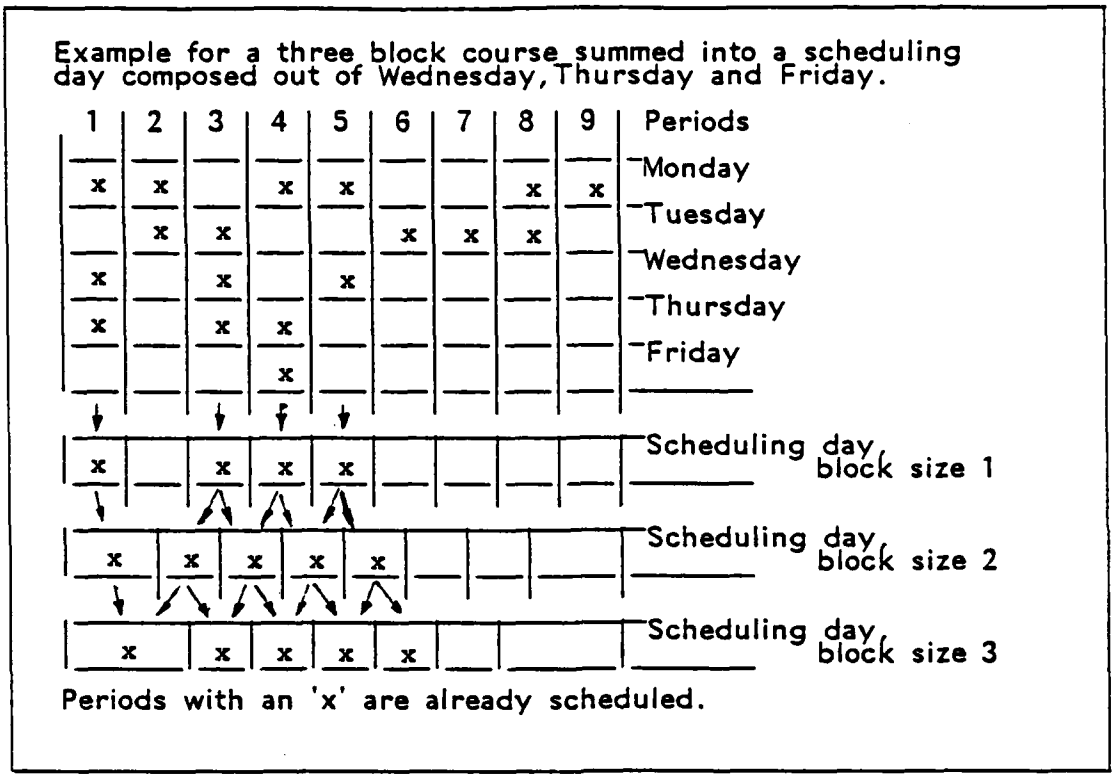



Figure 5.4 Building of a Scheduling Day.

3. If a feasible solution is reached then stop, go back to basic heuristic with results.
4. If no feasible solution is reached and further different combinations of week days to built a scheduling day are possible then
   - create new scheduling day
   - go to step 2.
5. If no feasible solution is reached using all possible scheduling days then enter interactive routine.

For the NPS scheduling problem block sizes of 1,2 or 3 consecutive hours are possible. Figure 5.6 shows that with a block size of one, B = 9 scheduling periods are possible during a day, B = 8 scheduling periods for a block size of two and B = 6 periods for a block size of three. At a block size of three one scheduling period

**Objective**

$$\underset{x,y}{\text{MIN}} \sum_{p} \left( \sum_{f} c_{fp} x_{fp} + \sum_{s} c_{sp} y_{sp} \right)$$

**subject to**

$$\sum_{p} x_{fp} = 1 \quad \forall f \qquad \text{assign every faculty member once} \qquad (1)$$

$$\sum_{p} y_{sp} = 1 \quad \forall s \in S_C \qquad \text{assign every section once} \qquad (2)$$

$$x_{f_1 p} + x_{f_2 p} \leq 1 \quad \forall p \qquad f_1, f_2 \text{ one faculty member} \qquad (3)$$

$$r_C \sum_{f} x_{fp} \leq \sum_{s \in S_C} m_s y_{sp} \leq R_C \sum_{f} x_{fp} \quad \forall f,p \qquad \text{section between r and R students} \qquad (4)$$

$$x_{fp} = 0 \quad \forall p \in P_f \qquad \text{no infeasible periods allowed} \qquad (5)$$

$$y_{sp} = 0 \quad \forall p \in P_s \qquad \text{no infeasible periods allowed} \qquad (6)$$

## Notation Summary

**Subscripts**

| | | |
|---|---|---|
| f | Faculty member | $f = \{1\}$ or $\{1,2\}$ or $\{1,2,3\}$ or $\{1,2,3,4\}$ |
| p | Periods | $p = 1,\ldots,B$ |
| s | Section | $s \in S_C$ |
| C | Segment | as many as faculty members |

**Variables**

| | | |
|---|---|---|
| $x_{fp}$ | Faculty member | 1 if f is assigned to p; 0 otherwise |
| $y_{sp}$ | Section | 1 if s is assigned to p; 0 otherwise |

**Constants**

| | |
|---|---|
| $C_{fp}$ | ... Cost for assigning f to p |
| $C_{sp}$ | ... Cost for assigning s to p |
| $m_s$ | ... Number of students in s |
| $r_C$ | ... Lower bound on segment size |
| $R_C$ | ... Upper bound on segment size |
| $P_f$ | ... Set of infeasible periods for f |
| $P_s$ | ... Set of infeasible periods for s |
| $S_C$ | ... Set of sections in course |
| B | ... 6,8,9 for block sizes 3,2,1 respectively |

Figure 5.5 ILP for the Scheduling of One Course.

(hour 4,5,6) would block all three possible lunch hours. It is therefore not considered.

Using the numbering of periods at the different block sizes the ILP can be employed for all block sizes as shown in Figure 5.5 The example in Figure 5.4 demonstrates how the addition of days to a scheduling day leads to different numbers of idle periods for the respective block size.

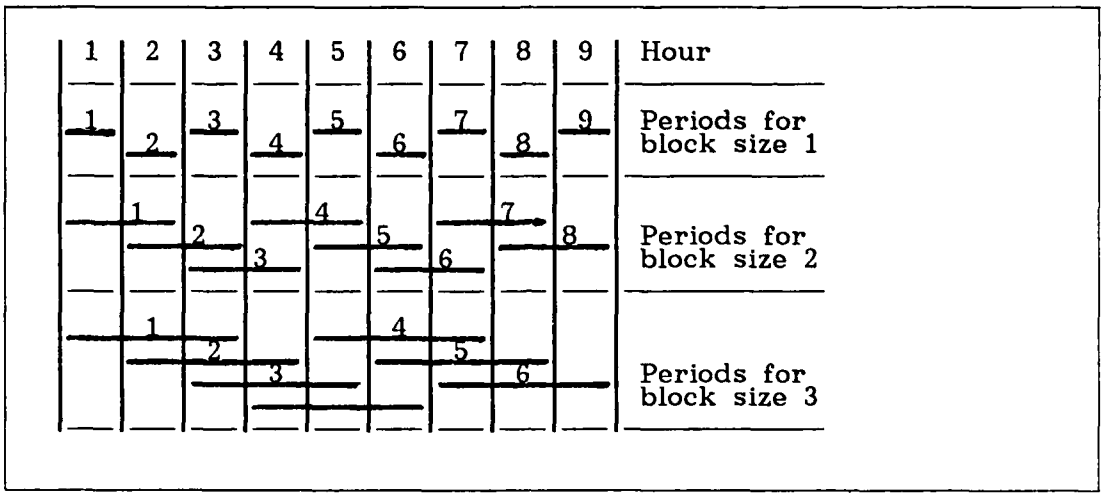



Figure 5.6 Periods of Different Block Sizes.

The variable $y_{sp}$ is determined by the section s and the period p of the scheduling day. It is generated only for idle periods (constraint (6)). The number of periods B is determined by the block size. The variables $x_{fp}$ are created if the faculty member f is idle at period p (constraint (5)).

b. Segments and Faculty Members

For the different cases of number of faculty members and number of segments, two measures have to be taken to use the ILP formulation of Figure 5.5: Create dummy faculty member variables and activate or deactivate constraint (3).

For each segment, one faculty member variable has to be created. These are naturally given in the cases (I),(III),(V) and

(VIII). While a course involving as many faculty members as segments is tried to schedule, constraint (3) of the ILP is deleted, because it is possible for all faculty members to teach during the same period.

Cases (II),(IV) and (VII) describe the situation of one faculty member teaching two segments. For this faculty member, two variables ($f_1$ and $f_2$) are created per period. As these two dummy faculty members can not teach during the same period the constraint set (3) has to be activated.

Case (VI) illustrates the possibility of two faculty members teaching two segments each. Here for each of the two faculty members two variables per period have to be created and the constraint set (3) has to be created for each faculty member.

c. The Objective Function

By means of the objective function, even more constraints and policy considerations can be brought into the scheduling ILP. When assigning low cost to a variable, the ILP will tend to bring this variable into the solution while minimizing the sum of the costs.

Students don't like to have gaps between their scheduled courses. To avoid gaps, periods adjoining already scheduled courses are assigned lower cost than to other periods. Faculty members request not to be scheduled at certain hours. Assigning high costs to those periods will drive the solution of the ILP to other periods. These costs must have a certain structure, however, to achieve the desired goal. The right proportion between low and high costs and costs for section and faculty variables will have to be determined during trial runs. A pricing policy which showed appropriate results during runs of an example course scheduling problem is presented in Table 12.

d. Back-to-Back Constraints

Back-to-back constraints can be accounted for within the pricing of the faculty member variables as long as two segments of one courses are not taught by one faculty member (Table 12). To incorporate these cases, additional sets of constraints need to be introduced.

TABLE 12

COST CONSIDERATIONS FOR THE COURSE ILP

| Kind | Event | Cost |
|---|---|---|
| Sections: | adjoining period | 1*c |
| | creating gap of 1 | $5*c*m_s$ |
| | creating gap of 2 | $10*c*m_s$ |
| | creating gap of 2 | $20*c*m_s$ |
| | otherwise | c |
| Faculty: | like to have | 1*c |
| | do not like to have | $10*c*R^{\&}$ |
| | adjoining period if not back-to-back | 20*c*R |
| | nonadjoining period if back-to-back | 10*c*R |
| | if no constraints | 1*c |

& R = (sum of students in course)/(number of segments)

c...the basic cost, i.e. '1'.

For the cases that a faculty member wants to teach both segments back-to-back, the constraints have to avoid that any combination other than adjoining periods is chosen. This can be reached by the following constraints:

$$x_{r_1 p} + x_{r_2(p+2)} \le 1 \quad \text{and} \quad x_{r_2 p} + x_{r_1(p+2)} \le 1 \quad \forall p \quad p = 1,\ldots,B-2 \qquad (7)$$

$$x_{r_1 p} + x_{r_2(p+3)} \le 1 \quad \text{and} \quad x_{r_2 p} + x_{r_1(p+3)} \le 1 \quad \forall p \quad p = 1,\ldots,B-3$$

$$\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots$$

$$x_{r_1 1} + x_{r_2 B} \le 1 \quad \text{and} \quad x_{r_2 1} + x_{r_1 B} \le 1$$

For the case that a faculty member requests not to have back-to-back teaching, the following set of constraints has to be included in the ILP:

$$x_{f_1 p} + x_{f_2 (p+1)} \leq 1 \quad \text{and} \quad x_{f_2 p} + x_{f_1 (p+1)} \leq 1 \quad \forall\, p \quad p = 1,\ldots,B-1 \tag{8}$$

e. Bounds on Segment Size

The lower and upper bound on the segment size ($r_C$, $R_C$) needed for constraint (4) is a function of the number of students in the course. As students must be distributed fairly evenly among the segments, the exact number of student per segment would be the ratio $R_{opt}$ = (students in course / number of segments). Suitable bounds for the ILP are:

$r_C = 0.9 * R_{opt}$, truncated

$R_C = 1.1 * R_{opt}$, rounded up

These bounds will have to be relaxed in cases were a few sections with relatively large number of students have to be distributed among some segments.

f. Assignment of Rooms

The assignment of a suitable room for a segment will be a problem only for courses with large segment sizes or lab-type courses as previous experience showed. Both types of 'problem courses' will be scheduled early within the process and thus reduce the amount of interaction with already scheduled courses. As a designated area for each course is defined, a choice of at most 30 rooms exists for any given course. The search for a room can therefore be performed by an exhaustive scan of the rooms in question. This task can, in principle be done as described for the final examination problem in section IV.B.3 with the exception that one and only one room per segment is allowed. Subroutines ROOMSUB and ROOM1 attached as part of Appendix A can be used with minor changes.

## 6. *The Manual Scheduling of a Course*

The manual scheduling of a course will be necessary if either a comment-type constraint exists or the automatic scheduling system fails to produce a feasible solution. In order to enable the human scheduler to schedule the course, a variety of information needs to be available to him/her. For a fast response the information should be brought to the screen in a condensed form rather than to a printer. This section discusses the kind of data necessary to provide the information for the scheduler and the problem of visualizing them, before sketching a system for the manual scheduling task.

### a. Necessary Data and Visual Presentation

The scheduler needs data about the course which is to be scheduled as well as the reason why the course could not be scheduled automatically. Depending on the reason, data about interfering courses or about rooms needs to be displayed, too. As a minimum, all data about the course as described in Table 7 the current timetable of all involved sections, F(*,*), and number of students in the section, NOSEC, from Table 9 the current timetable of the involved faculty members, P(*,*), as well as all of their constraints from Table 11 are necessary. Having these data, the scheduler knows the setup of the course. To be of any value to him/her, the information must be completly displayed in one piece. Without processing, however, it will not be possible to display all data even for a small course on one screen.

Figure 5.7 shows a possible way to display the data on a screen with 20 lines and 60 columns for one course. The left upper part of the screen presents the recent timetable for all sections, a matrix of nine periods and five weekdays. Each field consists of four possible single digit entries, a summary of the timetable entries of all sections. The most valuable single digit entry is the number of sections already scheduled at this period and day, enabling the scheduler to detect gaps or periods with few already scheduled sections. The other fields are open for experimenting, i.e. display of the best infeasible solution of the automatic system etc.

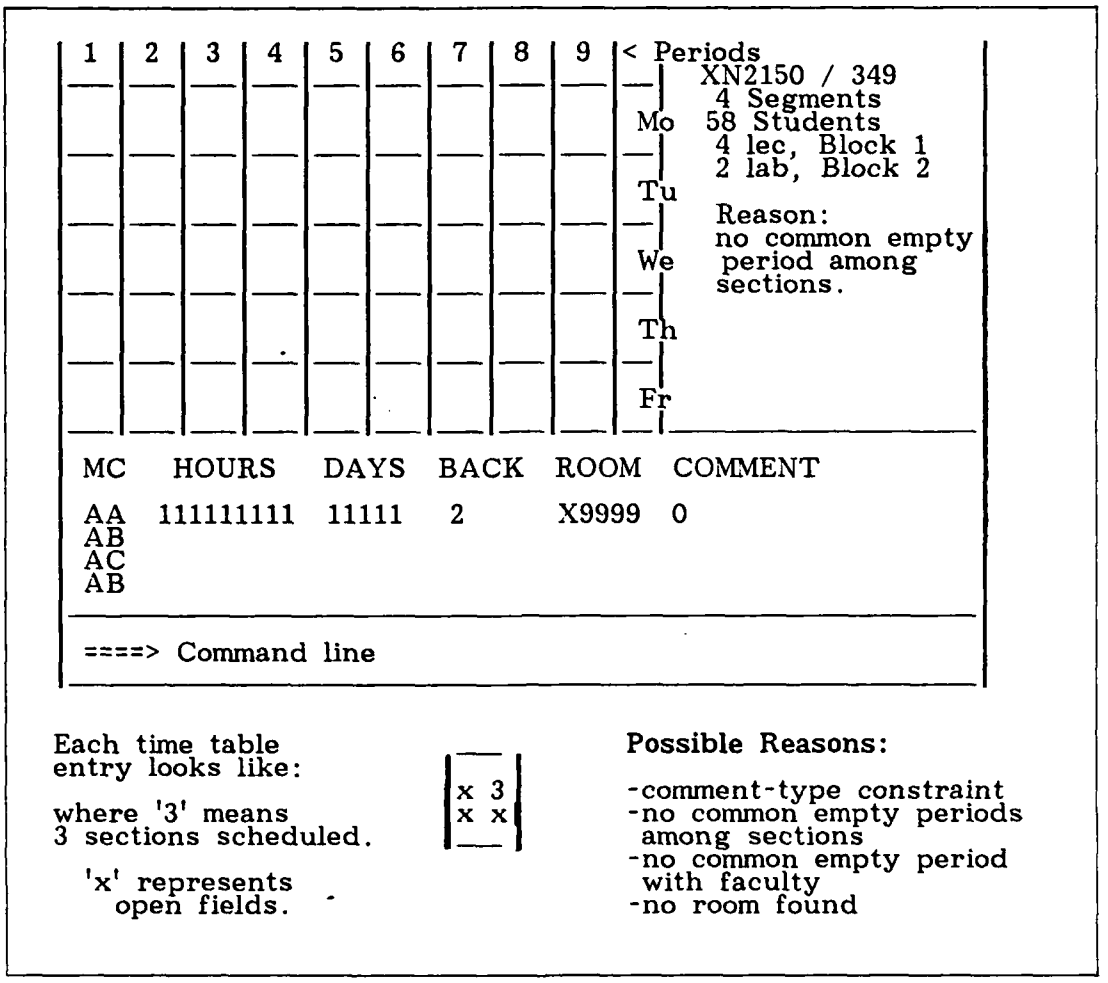



**Figure 5.7 Manual Scheduling Screen.**

The right upper area of the screen is reserved for course data common to all segments. Below these data a short comment about the reason for manual scheduling is displayed. Four rows are provided for entries of data about up to four faculty members following the timetabling format.

Although this screen provides a full overview of the current situation for some decisions, the scheduler needs more details. It will be necessary to know which course is interfering at a particular period or to display the timetable of a certain room. For this reason and for further scheduling purposes, the last line of the initial screen

is a command line. A command like **VIEW 7 WE,** would switch the display to detailed information about period 7 on Wednesday: The Wednesday timetable of the involved sections is displayed, giving the course data of already scheduled courses of period 7. This aids the scheduler's decision of which course he/she should reschedule, if no common empty period exists. The command **ROOM S210B,** would show the timetable of room S210b, if the problem should be an already filled room. **SECTIONS** calls a screen displaying all sections of the course with the number of students in the section and a 0/1 timetable of each section.

These three screens, together with the initial screen provide enough information for the scheduler for the manual scheduling task.

b. The Manual Process

The manual scheduling process, as proposed, uses a series of commands given from the command line of the screens described in the previous section to schedule the course in question. The kind of action taken depends on the reason for entering the manual system. Manual scheduling will always end with a return to the automatic scheduling routine. In this section some possible cases are discussed by looking at examples.

A full manual scheduling of a course, which is possible without rescheduling of any blocking course could be performed as described in the following example:

Given Data: 2 Segments
4 hours lecture, block size 1
2 faculty members: AB, XY
Comment: all computer science students professor XY.

Command: (1)**BUILD 2 SEGMENTS,** given from initial screen.
(2)**SEGMENT 1 ADD: XY,** assigns XY to segment 1.
(3)**SECTIONS,**brings up screen with sections.
(4)**SEGMENT 1 ADD: 99,203,409,533,611,** assigns the

sections to first segment.

(5) TIME 6MO,6TU,6WE,6FR,BLOCK=1, assigns periods.

(6) ROOM 50000, brings up all rooms in building 5 that are empty at period 6 and displays room number and number of places.

(7) SEGMENT 1 ROOM 51200, assigns room to segment.

(8) BACK, brings initial screen.

(9) SEGMENT 2 REST, assigns remaining faculty and sections to segment 2.

by similar commands as (3) and (5) to (8) segment 2 is scheduled.

(10) GO, checks the assignments for feasibility, updates the data base accordingly and gives control back to the automatic system.

A second possible case occurs when the summation into a scheduling day does not produce enough empty periods to schedule the segments the same hour every day.

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | Periods |
|---|---|---|---|---|---|---|---|---|---|
| & | | X | X | X | X | X | X | X | Monday |
| & | X | X | X | X | | | X | | Tuesday |
| X | X | & | | X | | | | X | Wednesday |
| X | | & | | | X | X | | | Thursday |
| X | | | X | | | X | X | | Friday |
| X | X | X | X | X | X | X | X | X | Scheduling day |

The extreme case of a 4 hour course is displayed here. No combination of week days produces a single empty period in the scheduling day. A possible solution would be to schedule (&) Monday and Tuesday the first period and Wednesday and Thursday the third period. This could be done using commands from the previous example.

The last example treats the hopeless case where the initial screen displays that at any period of the week, some section is already scheduled. The scheduler will have to unschedule one or more

courses in order to create empty periods. To do this in an effective way, the scheduler picks a period from the initial screen where the least sections are already scheduled. For example, he/she picks period 7 on Wednesday. Then, VIEW 7 WE displays the scheduled courses of the involved sections, lets say only one course: LS2112. COURSE LS2112 then displays the timetable of course LS2112. If unscheduling of LS2112 would not solve the problem, the scheduler goes BACK to the initial screen, otherwise the command UNSCHEDULE LS2112 unschedules this course and updates the data base. Course LS2112 is set in the priority file immediatly behind the course which is scheduled currently. The command SCHEDULE gives control back to the automatic system, which now tries to schedule this course with data from the changed data base.

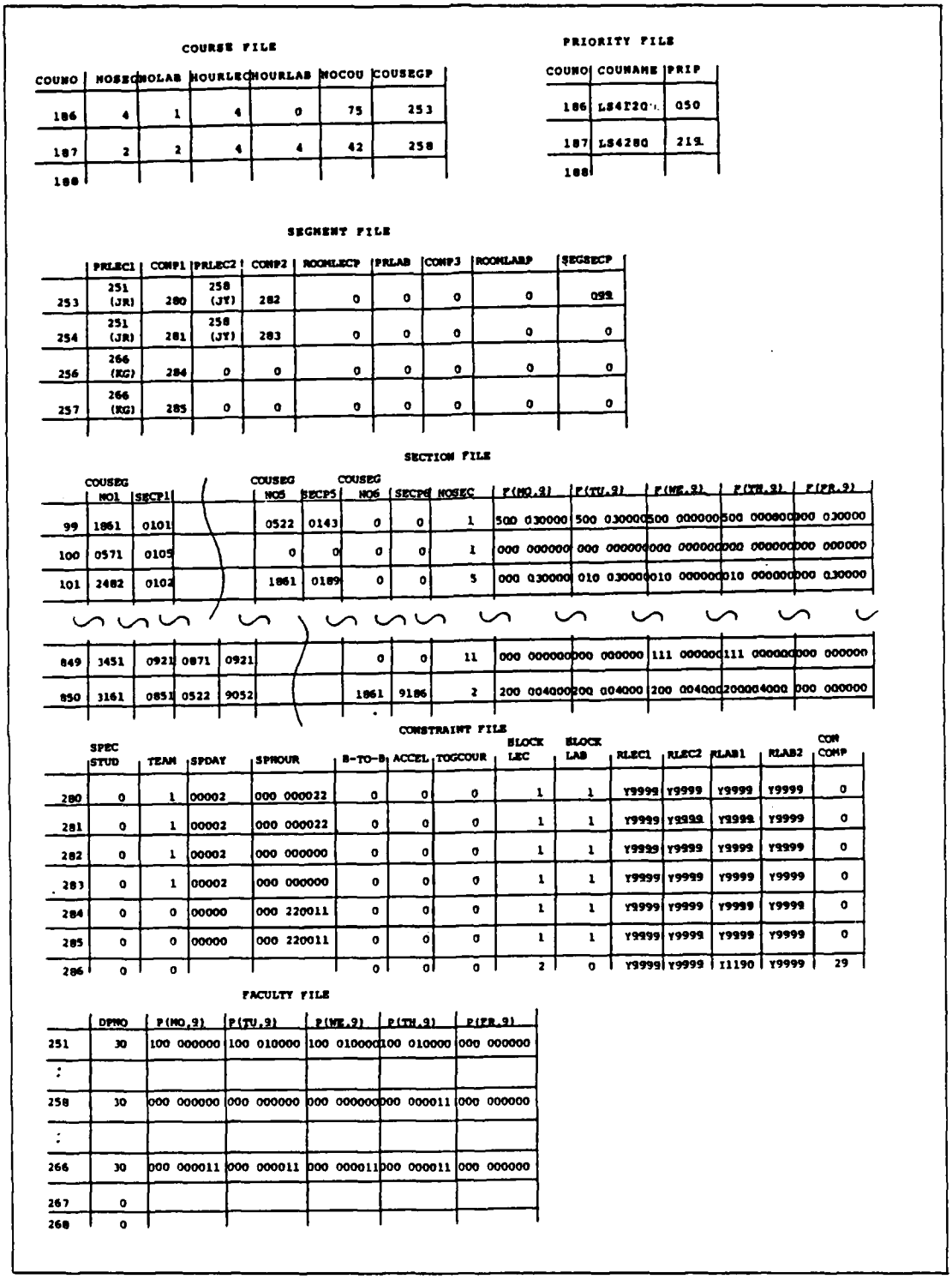
The described examples and the way to solve the problems resembles the actions currently performed by the scheduler. Experience with a future implemented system has to show whether the manual system outlined above is capable of handling the task.

## C. AN EXAMPLE

As an example, the course LS4120 is followed through the proposed system. Starting with the rough data, the entering of the information into the data base is performed and the course is scheduled by the automated system. Then the updated data base is described.

The course LS4120, course number 186, needs to be scheduled. The NPS catalog shows it with 4 lecture hours and no lab hour. 75 students are enrolled, grouped into 15 sections of different sizes. The decision of the department was to split the course into 4 segments. Two segments are taught in a team by professors JR and JY, the other two segments by professor KG. Entries from the constraint questionnaire for the three faculty members are shown in Table 13

After processing the rough data, the information is entered into the files of the data base. As the course has 75 students, 4 segments and team teaching, it receives a high priority. The scheduling of all courses with even higher priority is already performed and results are



COURSE FILE

| COUNO | NOSEC | NOLAB | HOURLEC | HOURLAB | NOCOU | COUSEGP |
|---|---|---|---|---|---|---|
| 186 | 4 | 1 | 4 | 0 | 75 | 253 |
| 187 | 2 | 2 | 4 | 4 | 42 | 258 |
| 188 | | | | | | |

PRIORITY FILE

| COUNO | COUNAME | PRIP |
|---|---|---|
| 186 | LS4120 | 050 |
| 187 | LS4280 | 219 |
| 188 | | |

SEGMENT FILE

| | PRLEC1 | CONF1 | PRLEC2 | CONF2 | ROOMLECP | PRLAB | CONF3 | ROOMLABP | SEGSECP |
|---|---|---|---|---|---|---|---|---|---|
| 253 | 251 (JR) | 280 | 258 (JY) | 282 | 0 | 0 | 0 | 0 | 092 |
| 254 | 251 (JR) | 281 | 258 (JY) | 283 | 0 | 0 | 0 | 0 | 0 |
| 256 | 266 (RG) | 284 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 257 | 266 (RG) | 285 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | | | | | | | | |

SECTION FILE

| | COUSEG NO1 | SECP1 | | | COUSEG NO5 | SECP5 | COUSEG NO6 | SECP6 | NOSEC | P(MO,9) | P(TU,9) | P(WE,9) | P(TH,9) | P(FR,9) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 99 | 1861 | 0101 | | | 0522 | 0143 | 0 | 0 | 1 | 500 030000 | 500 030000 | 500 000000 | 500 000000 | 000 030000 |
| 100 | 0571 | 0105 | | | 0 | 0 | 0 | 0 | 1 | 000 000000 | 000 000000 | 000 000000 | 000 000000 | 000 000000 |
| 101 | 2482 | 0102 | | | 1861 | 0189 | 0 | 0 | 5 | 000 030000 | 010 030000 | 010 000000 | 010 000000 | 000 030000 |
| 849 | 3451 | 0921 | 0871 | 0921 | | | 0 | 0 | 11 | 000 000000 | 000 000000 | 111 000000 | 111 000000 | 000 000000 |
| 850 | 3161 | 0851 | 0522 | 9052 | | | 1861 | 9186 | 2 | 200 004000 | 200 004000 | 200 004000 | 200004000 | 000 000000 |

CONSTRAINT FILE

| | SPEC STUD | TEAM | SPDAY | SPHOUR | B-TO-B | ACCEL | TOGCOUR | BLOCK LEC | BLOCK LAB | RLEC1 | RLEC2 | RLAB1 | RLAB2 | CON COMP |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 280 | 0 | 1 | 00002 | 000 000022 | 0 | 0 | 0 | 1 | 1 | Y9999 | Y9999 | Y9999 | Y9999 | 0 |
| 281 | 0 | 1 | 00002 | 000 000022 | 0 | 0 | 0 | 1 | 1 | Y9999 | Y9999 | Y9999 | Y9999 | 0 |
| 282 | 0 | 1 | 00002 | 000 000000 | 0 | 0 | 0 | 1 | 1 | Y9999 | Y9999 | Y9999 | Y9999 | 0 |
| 283 | 0 | 1 | 00002 | 000 000000 | 0 | 0 | 0 | 1 | 1 | Y9999 | Y9999 | Y9999 | Y9999 | 0 |
| 284 | 0 | 0 | 00000 | 000 220011 | 0 | 0 | 0 | 1 | 1 | Y9999 | Y9999 | Y9999 | Y9999 | 0 |
| 285 | 0 | 0 | 00000 | 000 220011 | 0 | 0 | 0 | 1 | 1 | Y9999 | Y9999 | Y9999 | Y9999 | 0 |
| 286 | 0 | 0 | | | 0 | 0 | 0 | 2 | 0 | Y9999 | Y9999 | I1190 | Y9999 | 29 |

FACULTY FILE

| | DPNO | P(MO,9) | P(TU,9) | P(WE,9) | P(TH,9) | P(FR,9) |
|---|---|---|---|---|---|---|
| 251 | 30 | 100 000000 | 100 010000 | 100 010000 | 100 010000 | 000 000000 |
| : | | | | | | |
| 258 | 30 | 000 000000 | 000 000000 | 000 000000 | 000 000011 | 000 000000 |
| : | | | | | | |
| 266 | 30 | 000 000011 | 000 000011 | 000 000011 | 000 000011 | 000 000000 |
| 267 | 0 | | | | | |
| 268 | 0 | | | | | |

Figure 5.8 Entries in Data Base before Scheduling.

TABLE 13

ENTRIES OF CONSTRAINT QUESTIONNAIRE FOR LS4120

| | TEAM | SPDAY | SPHOUR | BLOCK LEC | SEGNO |
|---|---|---|---|---|---|
| JR | Y | 00002 | 000000022 | 1 | 1,2 |
| JY | Y | 00002 | 000000000 | 1 | 1,2 |
| KG | N | 00000 | 000220011 | 1 | 3,4 |

read into the data base. Figure 5.8 reveals this situation. To prepare the data for the automatic scheduling routine, the scheduling days need to be created and the segments with team teaching need special consideration. For the team teaching segments, one faculty member, JJ, is created with the combined constraints of JR and JY. For the scheduling day Monday through Thursday is summed first as professor JR does not like Friday courses. Then two dummy variables for JJ and KG are created as both teach two segments. Faculty variables denote XA, XC for JJ and XB, XD for KG, section variables denote YA trough YO. The results of this preparation phase, including the assignment of costs for the objective function according to Table 12 are displayed in Figure 5.9.

From the entries of Figure 5.9, it is possible to generate the ILP. The bounds on the segment sizes were calculated to be (r,R) = (17,21). No variables need to be generated for period 5,8,9 as both faculty members are not available. The same is true for section variables at already scheduled periods. Constraint (3) of the ILP must be generated as faculty XA,XC are really one person (JJ), who cannot teach two segments at the same time.

This problem was solved using the linear programming package LINDO [Ref. 12]. The formulation and the solution is attached as Appendix J. The size and difficulty of this problem can be considered

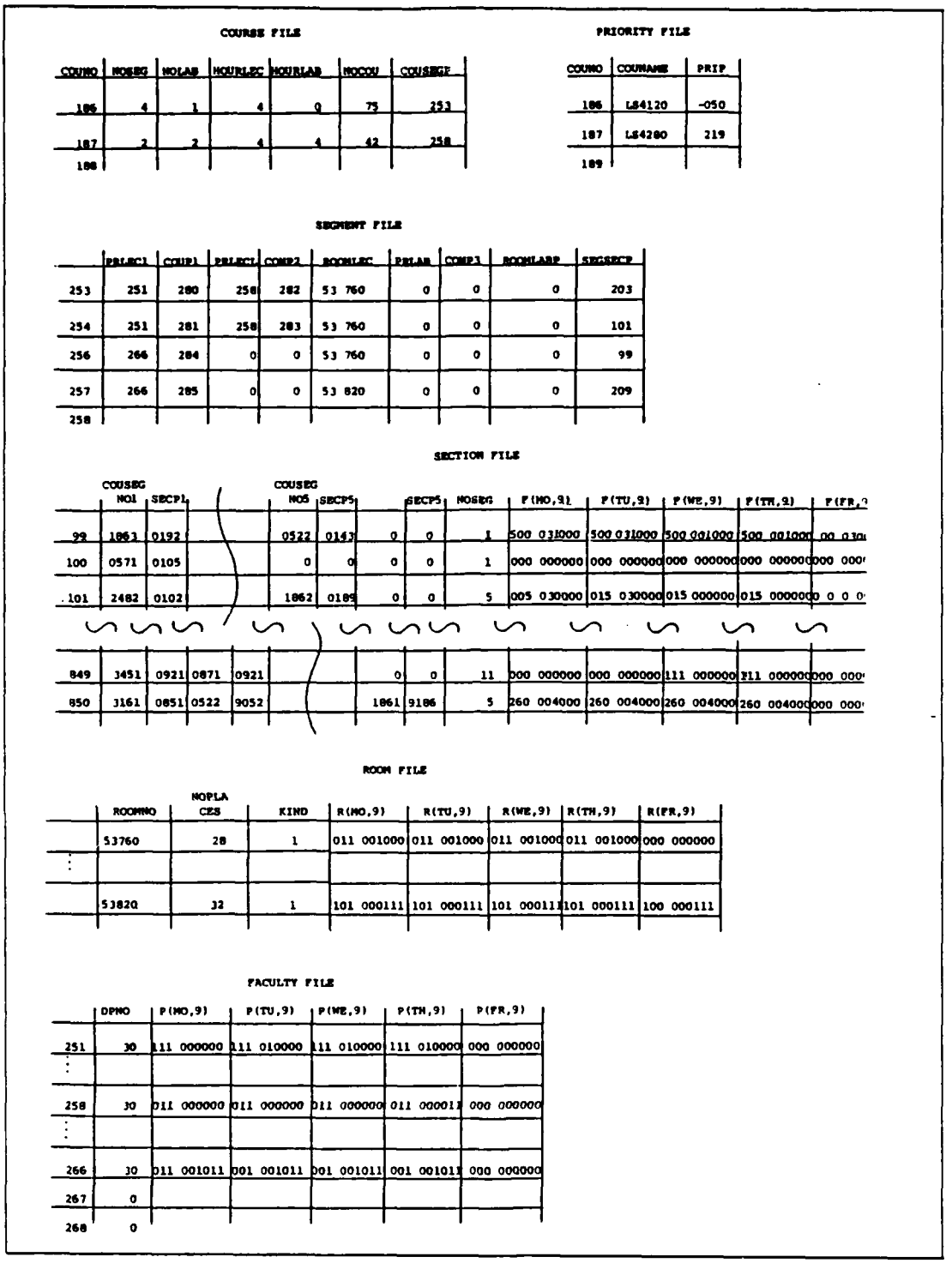
| Variables | $m_s$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| YA | 1 | x | 1 | 5 | 1 | x | 1 | 5 | 10 | 20 |
| YB | 5 | 1 | x | 1 | 1 | x | 1 | 25 | 50 | 100 |
| YC | 12 | 1 | x | 1 | 1 | x | 1 | 60 | 120 | 240 |
| YD | 9 | 1 | x | x | 1 | 45 | 90 | 180 | 1 | 1 |
| YE | 1 | x | 1 | 1 | x | 1 | 1 | x | 1 | 5 |
| YF | 1 | 1 | 1 | 20 | 10 | 5 | 1 | x | 1 | x |
| YG | 3 | x | x | 1 | 15 | 30 | 60 | 1 | 1 | 1 |
| YH | 6 | x | 1 | 1 | x | x | 1 | 30 | 60 | 120 |
| YI | 11 | 110 | 55 | 1 | x | x | 1 | 55 | 110 | 220 |
| YJ | 4 | 1 | 1 | 80 | 40 | 20 | 1 | x | 1 | x |
| YK | 3 | x | 1 | 15 | 1 | x | 1 | 15 | 30 | 60 |
| YL | 2 | x | 1 | 10 | 1 | x | 1 | 10 | 20 | 40 |
| YM | 8 | 1 | x | 1 | 40 | 40 | 1 | x | 1 | x |
| YN | 7 | x | 1 | x | 1 | 35 | 70 | 140 | 1 | 1 |
| YO | 2 | x | 1 | 10 | 10 | 1 | x | 1 | 10 | 20 |
| XA | - | x | 1 | 1 | 200 | x | 400 | 200 | x | x |
| XC | - | x | 1 | 1 | 200 | x | 400 | 200 | x | x |
| XB | - | 200 | 1 | 1 | x | x | 1 | 1 | x | x |
| XD | - | 200 | 1 | 1 | x | x | 1 | 1 | x | x |
| SUM | 75 | - | 20 | 34 | - | - | 21 | - | - | - |

cost: c=1, $R_{opt}$ = 20.

X: already scheduled or not available.

Figure 5.9 Prepared Input Data for ILP, Course LS4120.

typical for a course of this size which is scheduled with a high priority. For application purposes, the ILP will have to be generated in MPS format or some other standard format, and the ILP program must be connected to the rest of the scheduling program by an interface to reach the desired speed to solve the whole scheduling problem.



COURSE FILE

| | COUNO | NOSEG | NOLAB | HOURLEC | HOURLAB | NOCOU | COUSEGP |
|---|---|---|---|---|---|---|---|
| 186 | 4 | 1 | 4 | 0 | 75 | 253 |
| 187 | 2 | 2 | 4 | 4 | 42 | 258 |
| 188 | | | | | | |

PRIORITY FILE

| | COUNO | COUNAME | PRIP |
|---|---|---|---|
| 186 | LS4120 | -050 |
| 187 | LS4280 | 219 |
| 189 | | |

SEGMENT FILE

| | PRLEC1 | COUP1 | PRLEC1 | COMP2 | ROOMLEC | PRLAB | COMP3 | ROOMLABP | SEGSECP |
|---|---|---|---|---|---|---|---|---|---|
| 253 | 251 | 280 | 258 | 282 | 53 760 | 0 | 0 | 0 | 203 |
| 254 | 251 | 281 | 258 | 283 | 53 760 | 0 | 0 | 0 | 101 |
| 256 | 266 | 284 | 0 | 0 | 53 760 | 0 | 0 | 0 | 99 |
| 257 | 266 | 285 | 0 | 0 | 53 820 | 0 | 0 | 0 | 209 |
| 258 | | | | | | | | | |

SECTION FILE

| | COUSEG NO1 | SECP1 | | | COUSEG NO5 | SECP5 | | SECP5 | NOSEG | F(MO,9) | F(TU,9) | F(WE,9) | F(TH,9) | F(FR,9 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 99 | 1861 | 0192 | | | 0522 | 0143 | 0 | 0 | 1 | 500 031000 | 500 031000 | 500 001000 | 500 001000 | 00 030 |
| 100 | 0571 | 0105 | | | 0 | 0 | 0 | 0 | 1 | 000 000000 | 000 000000 | 000 000000 | 000 000000 | 000 000 |
| 101 | 2482 | 0102 | | | 1862 | 0189 | 0 | 0 | 5 | 005 030000 | 015 030000 | 015 000000 | 015 0000000 | 0 0 0 0 |
| 849 | 3451 | 0921 | 0871 | 0921 | | | 0 | 0 | 11 | 000 000000 | 000 000000 | 111 000000 | 111 000000 | 000 000 |
| 850 | 3161 | 0851 | 0522 | 9052 | | | 1861 | 9186 | 5 | 260 004000 | 260 004000 | 260 004000 | 260 004000 | 000 000 |

ROOM FILE

| | ROOMNO | NOPLACES | KIND | R(MO,9) | R(TU,9) | R(WE,9) | R(TH,9) | R(FR,9) |
|---|---|---|---|---|---|---|---|---|
| | 53760 | 28 | 1 | 011 001000 | 011 001000 | 011 001000 | 011 001000 | 000 000000 |
| | | | | | | | | |
| | 53820 | 32 | 1 | 101 000111 | 101 000111 | 101 000111 | 101 000111 | 100 000111 |

FACULTY FILE

| | DPNO | P(MO,9) | P(TU,9) | P(WE,9) | P(TH,9) | P(FR,9) |
|---|---|---|---|---|---|---|
| 251 | 30 | 111 000000 | 111 010000 | 111 010000 | 111 010000 | 000 000000 |
| | | | | | | |
| 258 | 30 | 011 000000 | 011 000000 | 011 000000 | 011 000011 | 000 000000 |
| | | | | | | |
| 266 | 30 | 011 001011 | 001 001011 | 001 001011 | 001 001011 | 000 000000 |
| 267 | 0 | | | | | |
| 268 | 0 | | | | | |

Figure 5.10 Entries in Data Base after Scheduling.

The solution produced by the ILP is feasible and therefore entered into the data base as seen in Figure 5.10. The course is marked as scheduled in the priority file and the control of the program takes the next course to schedule.

# VI. CONCLUSIONS

The purpose of this thesis was to automate the quarterly scheduling task at NPS. The main advantage of an automated and thus faster scheduling system is to avoid the standstill in updates within the registrar process. To reach this goal the scheduling process had to be analyzed, and two computerized scheduling systems had to be developed: The final examination system and the academic course scheduling system.

The final examination scheduling system has been developed and implemented. It consists of two programs to prepare the input data and schedule the exams. The preparation program runs on a batch system. The partly interactive scheduling system is based fully on a heuristic approach. Results with real data from quarter II/85 showed that the system is able to produce a complete and feasible examination schedule. The run time is between 90 to 120 CPU-seconds on a IBM 3033. Depending on the initial ordering of the input data a feasible solution may or may not be found at the first run; reordering of the input data PRIOR may be necessary.

Due to time constraints and the scope of the problem a scheduling system for academic courses could neither be fully developed nor implemented. The proposed approach, however, is likly to produce satisfactory results as it incorporates solution procedures to many of the unique features of the scheduling process at NPS. The scheduling of individual courses was tried using integer linear programming techniques. Many constraints could be incorporated in this approach and feasible results produced for example problems.

Further work on the academic course scheduling system is necessary to reach a reliable production system. The ILP approach to the single course scheduling problem showed promising results; it should be imbedded into the overall system. Extensive testing of any system will be necessary to reach acceptance at the user level.

Although the scheduling process will be automated in the near future, it will be essential to have a person with comprehensive knowledge about NPS and the scheduling permanently assigned to the process. This will enable individual treatment of requests, quick response to changes in the system, decrease the number of enrollment changes and ensure further high quality schedules.

# APPENDIX A

## FINAL PLC: PROGRAM LISTING

```
*PLC:  TIME=(6,0),NOSOURCE ,LINES=2000,PAGES=30;
FINAL: PROC OPTIONS (MAIN);
/*****************************************************************/
/*                                                               */
/*    THESIS    ROUTINE TO BUILT SCHEDULE FOR FINALS             */
/*    AUTHOR:                F I E G A S                        */
/*    INSTRUCTOR:            K. WOOD                            */
/*    DATE:          07      MAY 1985                           */
/*                                                               */
/*****************************************************************/
/*                                                               */
/*      EXEC TO RUN THE PROGRAM                                  */
/*                                                               */
/*                                                               */
/*                                                               */
/*****************************************************************/
/*&TRACE OFF*/
/*FILEDEF FIN DISK FINAL DTHE (RECFM FA LRECL 80 BLOCK 80 PERM*/
/*FILEDEF FAIL DISK FAIL DTHE (RECFM FA LRECL 80 BLOCK 80 PERM*/
/*FILEDEF PRIOR DISK PRIOR DATA (RECFM FA LRECL 500 BLOCK 500 */
/*FILEDEF ROOM DISK ROOM DATA (RECFM FA LRECL 80 BLOCK 80 PERM*/
/*FILEDEF DRUCKR DISK DRUCKR DATA (RECFM FA LRECL 80 BLOCK 80 */
/*FILEDEF DRUCKP DISK DRUCKP DATA (RECFM FA LRECL 80 BLOCK 80 */
/*EXEC PLC FINAL                                                */
/*X DRUCKR DATA                                                 */
/*FLIST                                                         */
/*&TRACE ON*/
/*****************************************************************/
/*                                                               */
/*  DECLARATION OF VARIABLES                                     */
/*                                                               */
/*                                                               */
/*                                                               */
/*****************************************************************/
DCL (N(400),PLACES(120),USE,F,R,TIMER(120,12),NQ,BACK(400),
     R1,R2,R3,R4,R6,REST(50),RE,EARL(20),NEXTONE,
     PTIME(0:702,12),COURSE(400),I,KBACK,TEMP1,T1   )FIXED(3);
DCL (FIRST,SECOND,SINGEL,HO,UR,HOU,R5,I5,V,V1,V2    )FIXED(2);
DCL (EOFROOM,EOFFIN,FI(400),EOFPRIO,COND(400),
     FREI,FOUND,FLAGCCC,DOWN,EQ,FEIN,OLDONE,PRFLAG )FIXED (1);
DCL (ROOMN(0:120),Y1,Y8,HOUR,ROOMP,ROOME,ITER,I4,J5)FIXED;
DCL (LOW,HIGH,X                                     ) FLOAT;
DCL (FA(400,4),YY                                   )CHAR (2);
DCL (CN(0:400),EARLY                                )CHAR (6);
DCL (XX                                             )CHAR (26);
DCL (D(400,2),HB(12,2),H1B(2),B1(200),TEMP2,NULLB   )BIT(200);
DCL (HH                                          )BIT (200) VAR;
DCL (D1,D2                                          )BIT(1);
/*****************************************************************/
/*                                                               */
/*      LABELS USED                                              */
/*                                                               */
/*                                                               */
/*                                                               */
/*****************************************************************/
```

```
/*    ALLES           STATEMENT LABEL CONSTANT                    */
/*    FILL            STATEMENT LABEL CONSTANT                    */
/*    INTER           STATEMENT LABEL CONSTANT                    */
/*    KEIN            STATEMENT LABEL CONSTANT                    */
/*    LOOK            STATEMENT LABEL CONSTANT                    */
/*    LOOK2           STATEMENT LABEL CONSTANT                    */
/*    NEI             STATEMENT LABEL CONSTANT                    */
/*    NEWR            STATEMENT LABEL CONSTANT                    */
/*    NX              STATEMENT LABEL CONSTANT                    */
/*    PART            STATEMENT LABEL CONSTANT                    */
/*    PRODRU          STATEMENT LABEL CONSTANT                    */
/*    SCHLUSSF        STATEMENT LABEL CONSTANT                    */
/*    SCHLUSSIN       STATEMENT LABEL CONSTANT                    */
/*    SCHLUSSR        STATEMENT LABEL CONSTANT                    */
/*    TRY             STATEMENT LABEL CONSTANT                    */
/*    WEITER2         STATEMENT LABEL CONSTANT                    */
/*    WEITER3         STATEMENT LABEL CONSTANT                    */
/*    WEITER4         STATEMENT LABEL CONSTANT                    */
/*    WEITER5         STATEMENT LABEL CONSTANT                    */
/*    ZZ1             STATEMENT LABEL CONSTANT                    */
/*    ZZ11            STATEMENT LABEL CONSTANT                    */
/*    ZZ2             STATEMENT LABEL CONSTANT                    */
/*    ZZ22            STATEMENT LABEL CONSTANT                    */
/*    ZZ3             STATEMENT LABEL CONSTANT                    */
/*    ZZ4             STATEMENT LABEL CONSTANT                    */
/*    ZZ44            STATEMENT LABEL CONSTANT                    */
/*    ZZ5             STATEMENT LABEL CONSTANT                    */
/*****************************************************************/
/*                                                               */
/*      VARIABLES USED                                           */
/*                                                               */
/*                                                               */
/*                                                               */
/*****************************************************************/
/*    BACK            (*)AUTOMATIC,ALIGNED,DECIMAL,FIXED(3,0)    */
/*    B1              (*)AUTOMATIC,UNALIGNED,BIT,STRING          */
/*    CN              (*)AUTOMATIC,UNALIGNED,CHARACTER,STRING    */
/*    COND            (*)AUTOMATIC,ALIGNED,DECIMAL,FIXED(1,0)    */
/*    COURSE          (*)AUTOMATIC,ALIGNED,DECIMAL,FIXED(3,0)    */
/*    D               (*,*)AUTOMATIC,UNALIGNED,BIT,STRING        */
/*    DOWN            AUTOMATIC,ALIGNED,DECIMAL,FIXED(1,0)       */
/*    D1              AUTOMATIC,ALIGNED,DECIMAL,FIXED(3,0)       */
/*    D1              AUTOMATIC,UNALIGNED,BIT,STRING             */
/*    D2              AUTOMATIC,ALIGNED,DECIMAL,FIXED(3,0)       */
/*    D2              AUTOMATIC,UNALIGNED,BIT,STRING             */
/*    E               (*)AUTOMATIC,UNALIGNED,BIT,STRING          */
/*    E               (*)AUTOMATIC,UNALIGNED,BIT,STRING          */
/*    EARL            (*)AUTOMATIC,ALIGNED,DECIMAL,FIXED(3,0)    */
/*    EARLY           AUTOMATIC,UNALIGNED,CHARACTER,STRING       */
/*    EOFFIN          AUTOMATIC,ALIGNED,DECIMAL,FIXED(1,0)       */
/*    EOFPRIO         AUTOMATIC,ALIGNED,DECIMAL,FIXED(1,0)       */
/*    EOFROOM         AUTOMATIC,ALIGNED,DECIMAL,FIXED(1,0)       */
/*    EQ              AUTOMATIC,ALIGNED,DECIMAL,FIXED(1,0)       */
/*    F               AUTOMATIC,ALIGNED,DECIMAL,FIXED(3,0)       */
/*    FA              (*,*)AUTOMATIC,UNALIGNED,CHARACTER,STRING  */
/*    FEIN            AUTOMATIC,ALIGNED,DECIMAL,FIXED(1,0)       */
/*    FI              (*)AUTOMATIC,ALIGNED,DECIMAL,FIXED(1,0)    */
/*    FIRST           AUTOMATIC,ALIGNED,DECIMAL,FIXED(2,0)       */
/*    FLAGCCC         AUTOMATIC,ALIGNED,DECIMAL,FIXED(1,0)       */
/*    FOUND           AUTOMATIC,ALIGNED,DECIMAL,FIXED(1,0)       */
/*    FRAND           (*)AUTOMATIC,ALIGNED,DECIMAL,FIXED(3,0)    */
/*    FREI            AUTOMATIC,ALIGNED,DECIMAL,FIXED(1,0)       */
/*    HB              (*,*)AUTOMATIC,UNALIGNED,BIT,STRING        */
/*    HH              AUTOMATIC,UNALIGNED,VARYING,BIT,STRING     */
/*    HIGH            AUTOMATIC,ALIGNED,DECIMAL,FLOAT(6)         */
```

```
/*    HO              AUTOMATIC,ALIGNED,DECIMAL,FIXED(2,0)      */
/*    HOU             AUTOMATIC,ALIGNED,DECIMAL,FIXED(2,0)      */
/*    HOUR            AUTOMATIC,ALIGNED,DECIMAL,FIXED(5,0)      */
/*    H1B             (*)AUTOMATIC,UNALIGNED,BIT,STRING         */
/*    I               AUTOMATIC,ALIGNED,DECIMAL,FIXED(3,0)      */
/*    ITER            AUTOMATIC,ALIGNED,DECIMAL,FIXED(5,0)      */
/*    I4              AUTOMATIC,ALIGNED,DECIMAL,FIXED(5,0)      */
/*    I5              AUTOMATIC,ALIGNED,DECIMAL,FIXED(2,0)      */
/*    J               AUTOMATIC,ALIGNED,DECIMAL,FIXED(5,0)      */
/*    J               AUTOMATIC,ALIGNED,DECIMAL,FIXED(5,0)      */
/*    J               AUTOMATIC,ALIGNED,DECIMAL,FIXED(5,0)      */
/*    J               AUTOMATIC,ALIGNED,DECIMAL,FIXED(5,0)      */
/*    J               AUTOMATIC,ALIGNED,BINARY,FIXED(15,0)      */
/*    J               AUTOMATIC,ALIGNED,DECIMAL,FIXED(3,0)      */
/*    J               AUTOMATIC,ALIGNED,DECIMAL,FIXED(3,0)      */
/*    J1              AUTOMATIC,ALIGNED,DECIMAL,FIXED(3,0)      */
/*    J5              AUTOMATIC,ALIGNED,DECIMAL,FIXED(5,0)      */
/*    K               PARAMETER,ALIGNED,DECIMAL,FIXED(5,0)      */
/*    K               PARAMETER,ALIGNED,DECIMAL,FIXED(5,0)      */
/*    K               PARAMETER,ALIGNED,DECIMAL,FIXED(5,0)      */
/*    KBACK           AUTOMATIC,ALIGNED,DECIMAL,FIXED(3,0)      */
/*    L               (*)AUTOMATIC,ALIGNED,DECIMAL,FIXED(3,0)   */
/*    LOW             AUTOMATIC,ALIGNED,DECIMAL,FLOAT(6)        */
/*    L1              AUTOMATIC,ALIGNED,DECIMAL,FIXED(3,0)      */
/*    L1              AUTOMATIC,ALIGNED,BINARY,FIXED(15,0)      */
/*    M1              (*)AUTOMATIC,ALIGNED,DECIMAL,FIXED(3,0)   */
/*    M2              (*,*)AUTOMATIC,UNALIGNED,BIT,STRING       */
/*    N               (*)AUTOMATIC,ALIGNED,DECIMAL,FIXED(3,0)   */
/*    NEXTONE         AUTOMATIC,ALIGNED,DECIMAL,FIXED(3,0)      */
/*    NQ              AUTOMATIC,ALIGNED,DECIMAL,FIXED(3,0)      */
/*    NULLB           AUTOMATIC,UNALIGNED,BIT,STRING            */
/*    N1              AUTOMATIC,ALIGNED,DECIMAL,FIXED(3,0)      */
/*    N2              AUTOMATIC,ALIGNED,DECIMAL,FIXED(3,0)      */
/*    N3              AUTOMATIC,ALIGNED,DECIMAL,FIXED(3,0)      */
/*    N4              AUTOMATIC,ALIGNED,DECIMAL,FIXED(3,0)      */
/*    N5              AUTOMATIC,ALIGNED,DECIMAL,FIXED(3,0)      */
/*    OLDONE          AUTOMATIC,ALIGNED,DECIMAL,FIXED(1,0)      */
/*    PLACES          (*)AUTOMATIC,ALIGNED,DECIMAL,FIXED(3,0)   */
/*    PRFLAG          AUTOMATIC,ALIGNED,DECIMAL,FIXED(1,0)      */
/*    PTIME           (*,*)AUTOMATIC,ALIGNED,DECIMAL,FIXED(3,0)*/
/*    R               AUTOMATIC,ALIGNED,DECIMAL,FIXED(3,0)      */
/*    RE              AUTOMATIC,ALIGNED,DECIMAL,FIXED(3,0)      */
/*    REST            (*)AUTOMATIC,ALIGNED,DECIMAL,FIXED(3,0)   */
/*    ROOME           AUTOMATIC,ALIGNED,DECIMAL,FIXED(5,0)      */
/*    ROOMN           (*)AUTOMATIC,ALIGNED,DECIMAL,FIXED(5,0)   */
/*    ROOMP           AUTOMATIC,ALIGNED,DECIMAL,FIXED(5,0)      */
/*    RR              AUTOMATIC,ALIGNED,DECIMAL,FIXED(3,0)      */
/*    RR              AUTOMATIC,ALIGNED,DECIMAL,FIXED(3,0)      */
/*    R1              AUTOMATIC,ALIGNED,DECIMAL,FIXED(3,0)      */
/*    R1              AUTOMATIC,ALIGNED,DECIMAL,FIXED(3,0)      */
/*    R1              AUTOMATIC,ALIGNED,DECIMAL,FIXED(3,0)      */
/*    R2              AUTOMATIC,ALIGNED,DECIMAL,FIXED(3,0)      */
/*    R2              AUTOMATIC,ALIGNED,DECIMAL,FIXED(3,0)      */
/*    R2              AUTOMATIC,ALIGNED,DECIMAL,FIXED(3,0)      */
/*    R3              AUTOMATIC,ALIGNED,DECIMAL,FIXED(3,0)      */
/*    R3              AUTOMATIC,ALIGNED,DECIMAL,FIXED(3,0)      */
/*    R4              (*)AUTOMATIC,ALIGNED,DECIMAL,FIXED(3,0)   */
/*    R4              (*)AUTOMATIC,ALIGNED,DECIMAL,FIXED(3,0)   */
/*    R4              AUTOMATIC,ALIGNED,DECIMAL,FIXED(3,0)      */
/*    R5              AUTOMATIC,ALIGNED,DECIMAL,FIXED(3,0)      */
/*    R5              AUTOMATIC,ALIGNED,DECIMAL,FIXED(3,0)      */
/*    R5              AUTOMATIC,ALIGNED,DECIMAL,FIXED(2,0)      */
/*    R6              AUTOMATIC,ALIGNED,DECIMAL,FIXED(3,0)      */
/*    R9              AUTOMATIC,ALIGNED,DECIMAL,FIXED(3,0)      */
/*    R9              AUTOMATIC,ALIGNED,DECIMAL,FIXED(3,0)      */
/*    SECOND          AUTOMATIC,ALIGNED,DECIMAL,FIXED(2,0)      */
```

```
/*   SINGEL          AUTOMATIC,ALIGNED,DECIMAL,FIXED(2,0)      */
/*   SUM             AUTOMATIC,ALIGNED,DECIMAL,FIXED(5,0)      */
/*   SUM             PARAMETER,ALIGNED,DECIMAL,FIXED(5,0)      */
/*   SUM             PARAMETER,ALIGNED,DECIMAL,FIXED(5,0)      */
/*   SUM             PARAMETER,ALIGNED,DECIMAL,FIXED(5,0)      */
/*   TEMP            (*)AUTOMATIC,ALIGNED,DECIMAL,FIXED(3,0)   */
/*   TEMP1           AUTOMATIC,ALIGNED,DECIMAL,FIXED(3,0)      */
/*   TEMP2           AUTOMATIC,UNALIGNED,BIT,STRING            */
/*   TIMER           (*,*)AUTOMATIC,ALIGNED,DECIMAL,FIXED(3,0)*/
/*   T1              AUTOMATIC,ALIGNED,DECIMAL,FIXED(3,0)      */
/*   UR              AUTOMATIC,ALIGNED,DECIMAL,FIXED(2,0)      */
/*   USE             AUTOMATIC,ALIGNED,DECIMAL,FIXED(3,0)      */
/*   V               AUTOMATIC,ALIGNED,DECIMAL,FIXED(2,0)      */
/*   V1              AUTOMATIC,ALIGNED,DECIMAL,FIXED(2,0)      */
/*   V2              AUTOMATIC,ALIGNED,DECIMAL,FIXED(2,0)      */
/*   W               (*)AUTOMATIC,UNALIGNED,CHARACTER,STRING   */
/*   W1              AUTOMATIC,UNALIGNED,CHARACTER,STRING      */
/*   W2              AUTOMATIC,UNALIGNED,CHARACTER,STRING      */
/*   W3              AUTOMATIC,UNALIGNED,CHARACTER,STRING      */
/*   X               AUTOMATIC,ALIGNED,DECIMAL,FLOAT(6)        */
/*   XX              AUTOMATIC,UNALIGNED,CHARACTER,STRING      */
/*   YY              AUTOMATIC,UNALIGNED,CHARACTER,STRING      */
/*   Y1              AUTOMATIC,ALIGNED,DECIMAL,FIXED(5,0)      */
/*   Y8              AUTOMATIC,ALIGNED,DECIMAL,FIXED(5,0)      */
/*****************************************************************/
/*                                                               */
/*    EXTERNAL FILES USED                                        */
/*                                                               */
/*                                                               */
/*                                                               */
/*****************************************************************/
/*   DRUCKP          FILE,STREAM,EXTERNAL                      */
/*   DRUCKR          FILE,STREAM,EXTERNAL                      */
/*   FAIL            FILE,STREAM,EXTERNAL                      */
/*   FIN             FILE,STREAM,EXTERNAL                      */
/*   PRIOR           FILE,STREAM,EXTERNAL                      */
/*   ROOM            FILE,STREAM,EXTERNAL                      */
/*****************************************************************/
/*                                                               */
/*     SUBROUTINES USED                                          */
/*                                                               */
/*                                                               */
/*                                                               */
/*****************************************************************/
/*   AGAIN           ENTRY,DECIMAL,FLOAT(6)                    */
/*   COMPLET         ENTRY,DECIMAL,FLOAT(6)                    */
/*   DRUCK           ENTRY,DECIMAL,FLOAT(6)                    */
/*   EIN             ENTRY,DECIMAL,FLOAT(6)                    */
/*   FINAL           ENTRY,DECIMAL,FLOAT(6)                    */
/*   GUIDE           ENTRY,DECIMAL,FLOAT(6)                    */
/*   LENK1           ENTRY,BINARY,FIXED(15,0)                  */
/*   LOESCH          ENTRY,BINARY,FIXED(15,0)                  */
/*   LOESCHP         ENTRY,BINARY,FIXED(15,0)                  */
/*   NEIGH           ENTRY,BINARY,FIXED(15,0)                  */
/*   NEIGH1          ENTRY,BINARY,FIXED(15,0)                  */
/*   PARTIAL         ENTRY,DECIMAL,FLOAT(6)                    */
/*   PROFSUB         ENTRY,DECIMAL,FLOAT(6)                    */
/*   ROOMSUB         ENTRY,DECIMAL,FLOAT(6)                    */
/*   ROOMS1          ENTRY,DECIMAL,FLOAT(6)                    */
/*   ROOMS2          ENTRY,DECIMAL,FLOAT(6)                    */
/*   ROOMS3          ENTRY,DECIMAL,FLOAT(6)                    */
/*   ROOMS4          ENTRY,DECIMAL,FLOAT(6)                    */
/*   SWITCH          ENTRY,DECIMAL,FLOAT(6)                    */
```

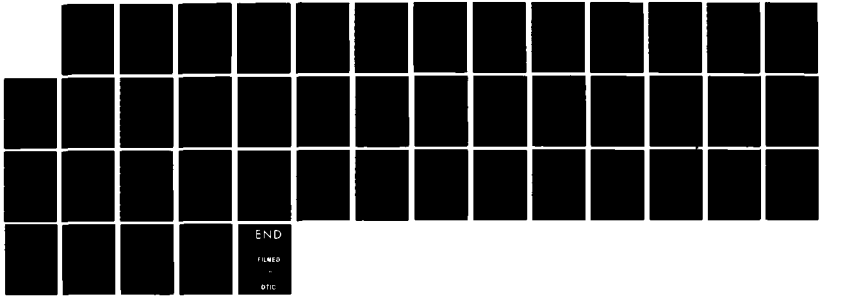
```
/*****************************************************************/
/*  EXTERNAL FILE DECLARATION                                    */
/*                                                               */
/*                                                               */
/*                                                               */
/*                                                               */
/*****************************************************************/
DCL PRIOR FILE STREAM;
DCL FIN   FILE STREAM;
DCL ROOM FILE STREAM;
DCL DRUCKR FILE STREAM;
DCL DRUCKP FILE STREAM;
DCL FAIL FILE STREAM;
/*****************************************************************/
/*                      \\\\V////                                */
/*    IF YOU WANT TO SEE THE PROGRAM WORK                        */
/*    SET THE PRFLAG=1; \\V//                                    */
/*                       \V/                                     */
/*                        V                                      */
/*****************************************************************/
                    PRFLAG=0;
/*****************************************************************/
/*    SET ENDFILE CONDITIONS FOR THE INPUT FILES                 */
/*                                                               */
/*                                                               */
/*                                                               */
/*                                                               */
/*****************************************************************/
EOFROOM=0;
EOFFIN=0;
EOFPRIO=0;
ON ENDFILE (ROOM) EOFROOM=1;
ON ENDFILE (FIN) EOFFIN=1;
ON ENDFILE (PRIOR) EOFPRIO=1;
/*****************************************************************/
/*  SET INITIAL CONDITIONS                                       */
/*                                                               */
/*                                                               */
/*                                                               */
/*                                                               */
/*****************************************************************/
B1(1)='1'B;         /*ARRAY OF DISTINCT BIT STRINGS , EACH */
                    /* ENTRY HAS ONE '1' AND 199 '0'*/
CN(0)='------';     /*CREATES ---- ON OUTPUT*/
D='0'B;             /*BIT STRING FOR EACH COURSE*/
DOWN=3;             /*ON OFF FOR BUILDING CONSTRAINTS*/
EARL=0;             /*ARRAY WITH EARLY FINAL COURSES */
FOUND=0;            /*FLAG FOR ROOM FOUND    */
FREI=0;             /*FLAG FOR PROF IDLE     */
HIGH=2.10;          /*FACTOR FOR UPPER ROOM SIZE*/
LOW=1.49;           /*FACTOR FOR LOWER ROMM SIZE*/
NULLB='0'B;         /*BIT STRING USED FOR COMPARISONS*/
ROOMN=0;            /*ARRAY OF USABLE ROOMS */
T1=1;               /* USED FOR INTERMEDIAT STORAGE*/
V=0;                /* USED FOR BRANCHING      */
V1=1;               /* USED FOR BRANCHING */
V2=1;               /* USED FOR BRANCHING  */
XX='ABCDEFGHIJKLMNOPQRSTUVWXYZ';/*USED FOR PROF ID */
X=.5;               /*SEED FOR RAND NUMBER GENERATOR*/
/*****************************************************************/
/*  INPUT FILE ROOM:  ROOMN=ROOMNUMBER (F(5))                    */
/*                    USE= ALL LECTURE ROOMS GET TRANFERED       */
/*                    PLACE=NUMBER OF PLACES AVAILABLE           */
/*                                                               */
/*                                                               */
/*****************************************************************/
```

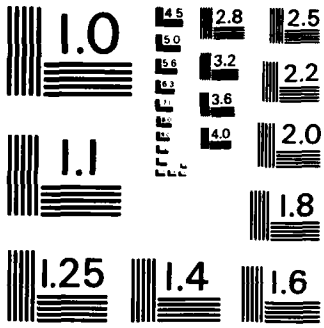
```
OPEN FILE (ROOM) INPUT;
DO I=1 TO 120;
      NEWR:
      GET FILE (ROOM) EDIT (ROOMN(I),USE,PLACES(I))
                (SKIP,F(5),X(2),F(2),X(7),F(2));
      IF EOFROOM=1 THEN GOTO SCHLUSSR;
      IF( USE>10)|(USE<1)|(USE=4) THEN GOTO NEWR;
/*****************************************************************/
/*  ROOMS, WHICH ARE NOT AVAILABLE DUE TO REFRESHER COURSES      */
/*  ARE FILTERED OUT                                             */
/*                                                               */
/*                                                               */
/*   THIS HAS TO BE SPECIFIED EVERY TIME                         */
/*****************************************************************/
      IF ROOMN(I)=43220    THEN GOTO NEWR;
      IF ROOMN(I)=43800    THEN GOTO NEWR;
      IF ROOMN(I)=71360    THEN GOTO NEWR;
END;
SCHLUSSR:
R=I-1;                                  /* NUMBER OF ROOMS*/

CLOSE FILE (ROOM);
/*****************************************************************/
/*    INPUT FILE FINAL DATA                                      */
/*   CN() COURSE NAME                                            */
/*   N() NUMBER OF STUDENTS IN COURSE                            */
/*   COND() LAB OR TEAMTEACH INDICATOR                           */
/*   FA(,) PROFS TEACHING       FI() '1' IF NO FINAL             */
/*****************************************************************/
OPEN FILE (FIN) INPUT;
DO I=1 TO 400;
    GET FILE (FIN) EDIT
   (CN(I),N(I),COND(I),FA(I,1),FA(I,2),FA(I,3)  ,FA(I,4),FI(I))
       (SKIP,X(8),A(6),X(5),F(3),X(12),F(1),X(4),A(2),X(3),A(2)
                                ,X(3),A(2),X(3),A(2),X(3),F(1));
      IF EOFFIN=1 THEN GOTO SCHLUSSF;
END;
SCHLUSSF:
F=I-1;                          /* # OF COURSES */
CLOSE FILE (FIN);
/*****************************************************************/
/*   INPUT PRIOR DATA                                            */
/*   COURSE() COURSE NUMBER                                      */
/*   D(,) BIT INDICATOR FOR EACH COURSE                          */
/*                                                               */
/*                                                               */
/*****************************************************************/
OPEN FILE (PRIOR) INPUT;
DO I=1 TO 400;
      GET FILE (PRIOR) EDIT (COURSE(I),D(I,1),D(I,2))
                (SKIP,F(3),X(2),B(200),B(200));
      IF EOFPRIO=1 THEN GOTO SCHLUSSIN;
END;
SCHLUSSIN:
CLOSE FILE (PRIOR);
/*****************************************************************/
/*   FILL THE ARRAY B1 WITH 200 DISTINCT BIT STRINGS             */
/*        EACH ONE '1' AND 199 '0'.                              */
/*                                                               */
/*                                                               */
/*                                                               */
/*****************************************************************/
HH='0'B;
DO I=2 TO 200;
   B1(I)=(HH || '1'B);
   HH =HH || '0'B;          /*USED TO CREATE THE B1() */
END;
```

AD-A161 396 THE NAVAL POSTGRADUATE SCHOOL SCHEDULING SYSTEM: A HEURISTIC APPROACH(U) NAVAL POSTGRADUATE SCHOOL MONTEREY CA D W FIEGAS SEP 85 2/2

UNCLASSIFIED F/G 12/1 NL






MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

```
/*****************************************************************/
/*    STAMP OUT UNSCHEDULED COURSES                              */
/*              AND COURSES WITHOUT FINAL.                       */
/*                                                               */
/*                                                               */
/*                                                               */
/*****************************************************************/
TRY:
IF PRFLAG=1 THEN
PUT SKIP(3) LIST ('','ANOTHER COMPLETE TRY');
PTIME=0;          /*ARRAY HOUR VS PROF, ENTRIES ARE COURSE #*/
TIMER=0;/*ARRAY HOUR VS ROOM, ENTRIES ARE COURSE #*/
HB='0'B;
DO Y8=1 TO F;
   BACK(COURSE(Y8))=Y8;       /*LINK BETWEEN FIN INPUT DATA  */
                              /*AND PRIOR INPUT DATA         */
   IF FI(COURSE(Y8))=1 | FA(COURSE(Y8),1)='00'THEN DO;
        IF COURSE(Y8)<201 THEN DO;
            D(*,1)=D(*,1)&(¬B1(COURSE(Y8)));
        END;
        ELSE DO;
            D(*,2)=D(*,2)&(¬B1(COURSE(Y8)-200));
        END;
        COURSE(Y8)=-1*COURSE(Y8);    /*NEGATIVE WHEN SCHEDULED*/
    END;
END;
/*****************************************************************/
/*                                                               */
/*     MAIN CONTROL OF PROGRAM                                   */
/*                                                               */
/*                                                               */
/*                                                               */
/*****************************************************************/
DOWN =3;                              /* HONOR BUILDING BOUNDS*/
IF V>1 THEN DO;                     /*NOT FOR THE FIRST TIME */
   CALL LENK1;          /*SCHEDULE EARLY FINALS AND NOT    */
END;                      /*SCHEDULED COURSES FIRST         */
ELSE DO;
   PUT SKIP(1) LIST ('PUT COURSES WITH EARLY FINALS');
   CALL EIN;     /*INPUT EARLY FINALS  */
   V=2;/*INDICATES AT LEAST SECOND RUN*/
END;
PART:
CALL PARTIAL;    /*SCHEDULE ALL */
CALL COMPLET;     /*CHECK FOR NOT SCHEDULED COURSES */
IF FLAGCCC=0 THEN DO;     /* IS '0' WHEN ALL COURSES SCHEDULED*/
   CALL DRUCK;      /*PRINT OUTPUT IN EXTERNAL FILES */
   STOP;
END;
IF (RE>F/22) & (V1<6) THEN DO;/*RE=# OF NOT SCHEDULED COURSES*/
   V1=V1 +1;     /*COUNTER */
   LOW=LOW*.98;           /*MAKES IT GRADUALY EASIER    */
   HIGH=HIGH*1.0;
   V=2;
   DO I=1 TO F;
        COURSE(I)=ABS(COURSE(I)); /*RESET SCHEDULE-INDICATOR*/
   END;
   GOTO TRY;         /*SCHEDULE ALL AGAIN */
END;
IF RE> F/18 THEN DO;
   GOTO INTER;      /*TOO MANY COURSES NOT SCHEDULED */
END;
NEI:
CALL NEIGH;       /*SELECTIVE RESCHEDULING  */
CALL COMPLET;
```

```
IF FLAGCCC=0 THEN DO;
   PUT SKIP LIST (' FEASIBLE SCHEDULE FOUND');
   CALL DRUCK;
   STOP;
END;
V2=V2+1;          /*COUNTER */
LOW=1.2;        /*RELAX BOUNDS*/
 HIGH=5.0;
IF V2< 6 THEN GOTO NEI;
IF V=3 THEN GOTO INTER;
V=3;
V2=2;
CALL AGAIN;       /*WHOLESALE RESCHEDULING  */
GOTO NEI;
/*****************************************************************/
/*   INTERACTIVE  CONTROL OF THE PROGRAM                         */
/*  (THERE IS A POSSIBILITY TO SCHEDULE SELECTIVE                */
/*   ONLY ONE COURSE: ANSWER THE QUESTION:                       */
/*   'TRY AGAIN ??????' WITH  2                                  */
/*                                                               */
/*****************************************************************/
INTER:
V=2;
PUT SKIP(2) LIST ('','NOT SCHEDULED UP TIL NOW');
PUT SKIP LIST ('NUMBER','COURSE','# OF STUDENTS');
DO Y8=1 TO RE;     /*PRINT NOT SCHEDULED COURSES TO SCREEN */
        I=BACK(REST(Y8));
        PUT SKIP LIST (COURSE(I),CN(COURSE(I)),N(COURSE(I)));
END;
   PUT SKIP LIST
   ('TRY AGAIN ??????:  1=YES  3=PRINT-AND-STOP  4= STOP ');
   GET EDIT (ITER)(COL(1),F(1));
  IF ITER=4 THEN DO;
  DO I=1 TO 12 WHILE (PRFLAG = 1);
  PUT SKIP LIST (I,HB(I,1),HB(I,2));
  END;
  STOP;
  END;
IF ITER=3 THEN DO;
   LOW=1.2;
   HIGH=8.0;
   CALL NEIGH;
   CALL COMPLET;
   IF FLAGCCC=0 THEN DO;
        PUT SKIP LIST (' FEASIBLE SCHEDULE FOUND');
        CALL DRUCK;
        STOP;
   END;
   ELSE DO;
/*****************************************************************/
/*                                                               */
/*   PRINT NOT SCHEDULED COURSES IN EXTERNAL FILE 'FAIL'         */
/*                                                               */
/*                                                               */
/*                                                               */
/*****************************************************************/
       OPEN FILE (FAIL) OUTPUT;
       PUT FILE (FAIL) SKIP LIST ('HOUR INDICATORS');
       DO I=1 TO 12;
          PUT FILE (FAIL) SKIP LIST (I,HB(I,1),HB(I,2));
       END;
       PUT FILE (FAIL) SKIP(2) LIST ('NOT SCHEDULED COURSES');
       PUT FILE (FAIL) SKIP (2)LIST
            ('#','COURSE #','COURSE');
       DO I=1 TO RE;
          PUT FILE (FAIL) SKIP LIST (I,REST(I),CN(REST(I)));
```

```
        END;
        CLOSE FILE (FAIL);
        CALL DRUCK;
        STOP;
    END;
END;
    PUT SKIP LIST ('SET LOW BOUND FOR PLACES, NOW',LOW);
    GET EDIT (LOW)(COL(1),F(3,1));
    PUT SKIP LIST('LOW SET TO:',LOW);
    PUT SKIP LIST ('SET HIGH BOUND FOR PLACES, NOW',HIGH);
    GET EDIT (HIGH)(COL(1),F(4,1));
    PUT SKIP LIST('HIGH SET TO:',HIGH);
    DOWN=3;
/*****************************************************************/
/*  THIS PART IS ONLY USED WHEN ITER = 2                         */
/*                                        IN TEST RUNS           */
/*                                                               */
/*                                                               */
/*                                                               */
/*****************************************************************/
    IF ITER=2 THEN DO;
        PUT SKIP LIST ('WITH PARTIAL OPTION:');
        PUT SKIP LIST
        ('YOU WANT TO SPECIFY THE BUILDING BOUNDS? YES=0 NO=3');
        GET EDIT (DOWN) (COL(1),F(1));
        IF DOWN¬=0 THEN GOTO KEIN;
        PUT SKIP LIST ('10000=BULLARD');
        PUT SKIP LIST ('20000=HALLIGAN');
        PUT SKIP LIST ('40000=INGERSOL');
        PUT SKIP LIST ('60000=ROOT');
        PUT SKIP LIST ('70000=SPANNAGEL');
        PUT SKIP LIST ('LOWER BUILDING NUMBER:');
        GET EDIT (ROOMP)(COL(1),F(5));
        PUT SKIP LIST ('HIGHER BUILDING NUMBER:');
        GET EDIT (ROOME)(COL(1),F(5));
         DOWN=2;
         GOTO PART;
KEIN:
        PUT SKIP LIST
         (' ','SET THE BUILDING CONSTRAINTS OFF, 0=YES  3=NO');
        GET EDIT (DOWN)(COL(1),F(1));
        GOTO PART;
    END;
    ELSE DO;
/*****************************************************************/
/*                                                               */
/*                                                               */
/*     ALL OVER AGAIN: SORT PRIOR NEW AND START AGAIN            */
/*                                                               */
/*                                                               */
/*****************************************************************/
V1=1;
V2=1;
PUT SKIP LIST
   ('DO YOU WANT TO SET THE EARLY FINALS NEW?  Y=1, N=2');
GET EDIT (V)(COL(1),F(1));
PUT SKIP LIST
    ('DO YOU WANT THE COURSES SORTED RANDOMLY? Y=1  N=3');
GET EDIT (ITER) (COL(1),F(1));
    IF ITER = 1 THEN DO;
        CALL SWITCH;  /*RESORTS THE THE COURSE ARRAY IN A */
        V=0;           /* SEMI RANDOM FASION */
        GOTO TRY;
    END;
    CALL COMPLET;
    DO I=1 TO F;
```

```
        COURSE(I)=ABS(COURSE(I));
    END;
GOTO TRY;
END;
/*****************************************************************/
/*                                                               */
/*              S U B R O U T I N S                              */
/*                                                               */
/*                                                               */
/*                                                               */
/*****************************************************************/
/*****************************************************************/
/*               WHOLESALE RESCHEDULING                          */
/*    RESCHEDULING OF COURSES WHICH ARE ALREADY SCHEDULED        */
/*                                                               */
/*        SUBROUTINE AGAIN                                       */
/*                                                               */
/*****************************************************************/
AGAIN: PROC;
DCL (TEMP(120),N1,N2,N3,N4,N5) FIXED (3);
IF PRFLAG=1 THEN
PUT SKIP LIST ('','','ENTER AGAIN');
DO N1=1 TO 8;
   N4=1;
   TEMP=0;
   DO N2=1 TO R;
        IF TIMER(N2,N1)=0 THEN GOTO NX;
        DO N5=1 TO N4;
             IF TIMER(N2,N1)=TEMP(N5) THEN GOTO NX;
        END;
        DO N5=1 TO 20 WHILE (N1<3);
             IF TIMER(N2,N1)=EARL(N5) THEN GOTO NX;
        END;
        Y8=BACK(TIMER(N2,N1));
        COURSE(Y8)=ABS(COURSE(Y8));
        DO N3=1 TO N1-1,N1+1 TO 8;
             HOUR=N3;
             HOU=N3;
             HO=N3;
             IF MOD(HOUR,2)=0 THEN HO=N3-1;
             CALL GUIDE;
             IF FREI=1 & FOUND=1 THEN DO;
                  HOUR=N1;
                  COURSE(Y8)=ABS(COURSE(Y8));
                  DO N5=N2 TO R;
                 IF TIMER(N5,N1)=COURSE(Y8) THEN TIMER(N5,N1)=0;
                  END;
                  CALL LOESCHP;
                 IF COURSE(Y8)<201 THEN
                 HB(HOUR,1)=(HB(HOUR,1) & (¬ B1(COURSE(Y8))));
                 ELSE
                HB(HOUR,2)=(HB(HOUR,2) &(¬B1(COURSE(Y8)-200)));
                  COURSE(Y8)=-1*COURSE(Y8);
                  GOTO NX;
             END;
            COURSE(Y8)=-1*COURSE(Y8);
        END;
        TEMP(N4)=COURSE(Y8);
        N4=N4+1;
NX:
   END;
END;
END AGAIN;
```

```
/*****************************************************************/
/*  PRINTOUT TO THE SCREEN:                                      */
/*       NOT SCHEDULED COURSE.                                   */
/*  REARANGE NOT SCHEDULED COURSES TO TOP OF PRIORITY LIST       */
/*                                                               */
/*      SUBROUTINE   COMPLET                                     */
/*****************************************************************/
PUT SKIP(2) LIST ('','NOT SCHEDULED UP TIL NOW');
PUT SKIP LIST ('NUMBER','COURSE','# OF STUDENTS');
        PUT SKIP LIST (COURSE(I),CN(COURSE(I)),N(COURSE(I)));
COMPLET: PROC;
RE=1;
REST=0;
FLAGCCC=0;
IF PRFLAG=1 THEN
PUT SKIP(2) LIST ('','NOT SCHEDULED UP TIL NOW');
IF PRFLAG=1 THEN
PUT SKIP LIST ('NUMBER','COURSE','# OF STUDENTS');
DO I=1 TO F;
BACK(ABS(COURSE(I)))=I;
    IF COURSE(I)>0 & FI(I)¬=1 THEN DO;
        IF PRFLAG=1 THEN
        PUT SKIP LIST (COURSE(I),CN(COURSE(I)),N(COURSE(I)));
             REST(RE)=COURSE(I);
             RE=RE+1;
        FLAGCCC=1;
    END;
END;
RE=RE-1;                          /* # OF NOT SCHEDULED COURSES */
RETURN;
END COMPLET;
/*****************************************************************/
/*      DEFINE COURSES WITH EARLY FINALS                         */
/*                                                               */
/*      SUBROUTINE  EIN                                          */
/*                                                               */
/*                                                               */
/*****************************************************************/
EIN: PROC;
I5=1;
DO I4=1 TO 20;
   PUT SKIP LIST ('PUT "000000" WHEN READY');
   PUT SKIP LIST (I4,'COURSE');
   GET EDIT (EARLY) (COL(1),A(6));
   IF SUBSTR(EARLY,1,1)='0' THEN DO;
       I5=I5-1;
       RETURN;
   END;
   PUT SKIP LIST (EARLY);
   DO J5=1 TO F;
      IF INDEX(EARLY,CN(ABS(COURSE(J5))))=1 THEN GOTO LOOK;
   END;
   GOTO LOOK2 ;
LOOK:
             EARL(I5)=ABS(COURSE(J5));
     PUT EDIT ('NUMBER','COURSE','SIZE')
             (SKIP(1),A(24),A(24),A(24));
     PUT SKIP LIST (ABS(COURSE(J5)),CN(ABS(COURSE(J5))),
                    N(ABS(COURSE(J5))));
             Y8= J5;
             HOUR=1;
           HOU=1;
           HO=1;
             CALL GUIDE;
                IF FOUND¬=1 THEN DO;
                    Y8= J5;
```

```
                    HOU=2;
                    HO=1;
                    HOUR=2;
                    CALL GUIDE;
                END;
          IF FOUND = 1 THEN DO;
              I5=I5+1;
             PUT SKIP LIST('SCHEDULED AT HOUR',HOUR);
          END;
          ELSE DO;
          PUT SKIP LIST
                ('NOT SCHEDULED BECAUSE ');
    IF FREI=9 THEN PUT LIST (' ALREADY SCHEDULED');
    IF FREI=8 THEN PUT LIST (' CONFLICTS IN FIRST TWO HOURS');
    IF FREI=0 THEN PUT LIST (' PROF BUISY IN FIRST TWO HOURS');
    IF FREI=0 THEN PUT LIST (' PROF BUISY IN FIRST TWO HOURS');
    IF FREI=1 & FOUND¬=1 THEN
                PUT LIST (' NO SIZE ROOM IN FIRST TWO HOURS');
          END;
LOOK2:
END;
I5=I5-1;
RETURN;
END EIN;
/*****************************************************************/
/*    SUBROUTINE LOOKING FOR POSSIBLE COURSES TO SCHEDULE      */
/*    WITHIN THIS HOUR.                                          */
/*                                                               */
/*        SUBROUTINE  GUIDE                                      */
/*                                                               */
/*****************************************************************/
GUIDE: PROC;
FREI=9;
FOUND=0;
IF COURSE(Y8)>0 THEN DO;
    FREI=8;
    H1B(1)=HB(HOU,1) & (¬D(Y8,1));
    D1=(H1B(1)=HB(HOU,1));
    H1B(2)=HB(HOU,2) & (¬D(Y8,2));
    D2=(H1B(2)=HB(HOU,2));
    IF D1&D2 THEN DO;
        I=Y8;
        CALL PROFSUB;
        IF FREI=0|FOUND¬=1 THEN DO;
            IF HO=1 THEN  HOUR=9;
            IF HO=3 THEN HOUR=10;
            IF HO=5 THEN HOUR=11;
            IF HO=7 THEN HOUR=12;
            H1B(1)=HB(HOUR,1) & (¬D(Y8,1));
            D1=(H1B(1)=HB(HOUR,1));
            H1B(2)=HB(HOUR,2) & (¬D(Y8,2));
            D2=(H1B(2)=HB(HOUR,2));
            IF D1&D2 THEN DO;
               I=Y8;
               CALL PROFSUB;
                 IF FOUND¬=1 | FREI=0 THEN DO;
                      HOUR=HOU;
                      RETURN;
                 END;
                 IF FREI=1 THEN DO;
                     IF COURSE(Y8)<201 THEN
                  HB(HOUR,1)=(HB(HOUR,1) | B1(COURSE(Y8)));
          ELSE HB(HOUR,2)=(HB(HOUR,2) | B1(COURSE(Y8)-200));
                 END;
                 HOUR=HOU;
            END;
```

```
        END;
        IF FREI=1 THEN DO;
            IF COURSE(Y8)<201 THEN
                   HB(HOU,1)=(HB(HOU,1) | B1(COURSE(Y8)));
            ELSE HB(HOU,2)=(HB(HOU,2) | B1(COURSE(Y8)-200));
            COURSE(Y8)=-1*COURSE(Y8);
        END;
    END;
END;
RETURN;
END GUIDE;
/*****************************************************************/
/*   PRINT THE HOUR INDICATORS AND NOT SCHEDULED COURSES         */
/*                                 INTO FILE    FAIL DATA        */
/*           SUROUTINE   DRUCK                                   */
/*                                                               */
/*                                                               */
/*****************************************************************/
DRUCK: PROC;
DCL(W(7),W1,W2                                  )CHAR (1);
DCL(W3                                          )CHAR (7);
W3=' ABCDEF';
W(1)='B';
W(2)='H';
W(3)='E';
W(4)='I';
W(5)='O';
W(6)='R';
W(7)='S';
         OPEN FILE (DRUCKR) OUTPUT;
         OPEN FILE (DRUCKP) OUTPUT;
PUT FILE (DRUCKR) EDIT
    ('     ROOM SCHEDULE FOR FINALS')(SKIP,A(50));
PUT FILE (DRUCKP) EDIT
    ('     FACULTY SCHEDULE FOR FINALS')(SKIP,A(50));
/*****************************************************************/
/*   PRINT THE ACHIEVED ROOM SCHEDULE                            */
/*                                                               */
/*                                                               */
/*                                                               */
/*                                                               */
/*****************************************************************/
PUT FILE (DRUCKR) EDIT
    ('ROOM','MO 1','MO 2','TU 1','TU 2','WE 1','WE 2'
      ,'TH 1','TH 2','MO 3','TU 3','WE 3','TH 3')
        (SKIP(2) ,X(3),A(5),X(3),
        X(2),A(6),
        X(2),A(6),
        X(2),A(6),
        X(2),A(6),
        X(2),A(6),
        X(2),A(6),
        X(2),A(6),
        X(2),A(6),
        X(6),A(6),
        X(2),A(6),
        X(2),A(6),
        X(2),A(6),SKIP);

DO I=1 TO R;
   ROOMP=ROOMN(I)/10;
   ROOMP=ROOMN(I)-ROOMP*10;
   ITER=ROOMN(I)/10000;
   T1=(ROOMN(I)-ITER*10000-ROOMP)/10;
   W1=W(ROOMN(I)/10000);
   W2= SUBSTR(W3,ROOMP+1,1);
```

```
   PUT FILE (DRUCKR) EDIT (W1,T1,W2,
   CN(ABS(TIMER(   I ,1))),
   CN(ABS( TIMER(   I ,2))),
   CN(ABS(TIMER(   I ,3))),
   CN(ABS(TIMER(   I ,4))),
   CN(ABS(TIMER(   I ,5))),
   CN(ABS(TIMER(   I ,6))),
   CN(ABS(TIMER(   I ,7))),
   CN(ABS(TIMER(   I ,8))),
   CN(ABS(TIMER(   I ,9))),
   CN(ABS(TIMER(   I ,10))),
   CN(ABS(TIMER(   I ,11))),
   CN(ABS(TIMER(   I ,12))))
        (SKIP,X(3),A(1),F(3),A(1),X(3),
        X(2),A(6),
        X(2),A(6),
        X(2),A(6),
        X(2),A(6),
        X(2),A(6),
        X(2),A(6),
        X(2),A(6),
        X(2),A(6),
        X(6),A(6),
        X(2),A(6),
        X(2),A(6),
        X(2),A(6));
END;
IF PRFLAG=0 THEN GOTO PRODRU;
PUT FILE (DRUCKR) EDIT
    ('     ROOM SCHEDULE FOR FINALS')(SKIP(5),A(50));
PUT FILE (DRUCKR) EDIT
    ('ROOM','MO 1','MO 2','TU 1','TU 2','WE 1','WE 2'
     ,'TH 1','TH 2','MO 3','TU 3','WE 3','TH 3')
       (SKIP(2) ,X(3),A(5),X(3),
        X(2),A(6),
        X(2),A(6),
        X(2),A(6),
        X(2),A(6),
        X(2),A(6),
        X(2),A(6),
        X(2),A(6),
        X(2),A(6),
        X(6),A(6),
        X(2),A(6),
        X(2),A(6),
        X(2),A(6),SKIP);
DO I=1 TO R;
   PUT FILE (DRUCKR) EDIT (ROOMN(I),
      (TIMER(   I ,1)),
      ( TIMER(   I ,2)),
       (TIMER(   I ,3)),
       (TIMER(   I ,4)),
       (TIMER(   I ,5)),
       (TIMER(   I ,6)),
       (TIMER(   I ,7)),
       (TIMER(   I ,8)),
       (TIMER(   I ,9)),
       (TIMER(   I ,10)),
       (TIMER(   I ,11)),
       (TIMER(   I ,12)))
        (SKIP,X(3),F(5),X(3),
        X(2),F(6),
        X(2),F(6),
        X(2),F(6),
        X(2),F(6),
        X(2),F(6),
```

```
        X(2),F(6),
        X(2),F(6),
        X(2),F(6),
        X(6),F(6),
        X(2),F(6),
        X(2),F(6),
        X(2),F(6));
END;
/*****************************************************************/
/*   PRINT THE ACHIEVED PROFESSOR SCHEDULE                       */
/*                                                               */
/*                                                               */
/*                                                               */
/*                                                               */
/*****************************************************************/
PRODRU:
PUT FILE (DRUCKP) EDIT
    ('PROF','MO 1','MO 2','TU 1','TU 2','WE 1','WE 2'
      ,'TH 1','TH 2','MO 3','TU 3','WE 3','TH 3')
        (SKIP(2) ,X(3),A(5),X(3),
        X(2),A(6),
        X(2),A(6),
        X(2),A(6),
        X(2),A(6),
        X(2),A(6),
        X(2),A(6),
        X(2),A(6),
        X(2),A(6),
        X(6),A(6),
        X(2),A(6),
        X(2),A(6),
        X(2),A(6),SKIP);
DO I=0 TO 675;
   Y1=MOD(I,26);
   YY=(SUBSTR(XX,((I-Y1)/26)+1,1))||(SUBSTR(XX,(Y1+1),1));
   PUT FILE (DRUCKP) EDIT(YY,
         CN( PTIME(I,1)),
         CN( PTIME(I,2)),
         CN( PTIME(I,3)),
         CN( PTIME(I,4)),
         CN( PTIME(I,5)),
         CN( PTIME(I,6)),
         CN( PTIME(I,7)),
         CN( PTIME(I,8)),
         CN( PTIME(I,9)),
         CN( PTIME(I,10)),
         CN( PTIME(I,11)),
         CN( PTIME(I,12)))
        (SKIP,X(3),A(2),X(3),
        X(2),A(6),
        X(2),A(6),
        X(2),A(6),
        X(2),A(6),
        X(2),A(6),
        X(2),A(6),
        X(2),A(6),
        X(2),A(6),
        X(6),A(6),
        X(2),A(6),
        X(2),A(6),
        X(2),A(6));
END;
         CLOSE FILE (DRUCKR);
         CLOSE FILE (DRUCKP);
RETURN;
END DRUCK;
```

```
/*****************************************************************/
/*  IN A NEW TRY, SCHEDULE EARLY FINALS AND FORMER NOT           */
/*       SCHEDULED COURSES FIRST                                 */
/*                                                               */
/*     SUBROUTINE LENK1                                          */
/*                                                               */
/*****************************************************************/
LENK1: PROC;
IF PRFLAG=1 THEN
PUT SKIP LIST ('ENTER LENK1');
DO R3=1  TO I5;
   Y8=BACK(EARL(R3));
   HOUR=1;
   HOU=1;
   HO=1;
   CALL GUIDE;
   IF FOUND¬=1 | FREI¬=1 THEN DO;
        Y8= BACK(EARL(R3));
        HOU=2;
        HO=1;
        HOUR=2;
        CALL GUIDE;
   IF FOUND ¬= 1 THEN DO;
        PUT SKIP LIST
              ('NOT SCHEDULED BECAUSE ');
   IF FREI=9 THEN PUT LIST (' ALREADY SCHEDULED');
   IF FREI=8 THEN PUT LIST (' CONFLICTS IN FIRST TWO HOURS');
   IF FREI=0 THEN PUT LIST (' PROF BUISY IN FIRST TWO HOURS');
   IF FREI=0 THEN PUT LIST (' PROF BUISY IN FIRST TWO HOURS');
   IF FREI=1 & FOUND¬=1 THEN
               PUT LIST (' NO SIZE ROOM IN FIRST TWO HOURS');
   END;
   END;
END;
DO HO=1,3,5,7;
   DO UR=0 TO 1;
   HOUR=HO+UR;
   HOU=HOUR;
        DO R3=1 TO RE;
             Y8=BACK(REST(R3));
             CALL GUIDE;
        END;
   END;
END;
RETURN;
END LENK1;
/*****************************************************************/
/*  SUBROUTINE FOR DELETION OF COURSES FROM ROOM SCHEDULE        */
/*                                                               */
/*                                                               */
/*       SUBROUTINE LOESCH                                       */
/*                                                               */
/*****************************************************************/
LOESCH: PROC;
   DCL (J                          ) FIXED (3);
   DO J=1 TO 120;
        IF TIMER(J,HOUR)=COURSE(I) THEN TIMER (J,HOUR)=0;
   END;
   RETURN;
END LOESCH;
/*****************************************************************/
/*  SUBROUTINE FOR DELETION OF COURSES FROM PROF SCHEDULE        */
/*                                                               */
/*                                                               */
/*       SUBROUTINE LOESCHP                                      */
/*                                                               */
/*****************************************************************/
```

```
LOESCHP: PROC;
   DCL (J                          ) FIXED (3);
   DO J=1 TO 702;
        IF PTIME(J,HOUR)=COURSE(I) THEN PTIME (J,HOUR)=0;
   END;
   RETURN;
END LOESCHP;
/*****************************************************************/
/*  SCHEDULING OF NOT SCHEDULED COURSES BY RESCHEDULING          */
/*  OF OPPOSING ALREADY SCHEDULED COURSES                        */
/*                                                               */
/*    SUBROUTINE NEIGH:                                          */
/*                                                               */
/*****************************************************************/
NEIGH: PROC;
DCL(E(2)                        )BIT (200);
DCL (R4(6),R9,RR,R1,R2,R5,R3  )FIXED (3);
DO R3= 1 TO RE;
   DO R1=1 TO 8;
        RR=1;
        R4=0;
        E(1)=HB(R1,1) & D(BACK(REST(R3)),1);
        E(2)=HB(R1,2) & D(BACK(REST(R3)),2);
        DO R2=1 TO 200;
          TEMP2= E(1) & B1(R2);
          IF TEMP2=B1(R2) THEN DO;
             IF  RR<= V2 THEN DO;
                  R4(RR)=R2;
                  RR=RR+1;
             END;
             ELSE DO;
                  GOTO ZZ2;
             END;
          END;
          TEMP2= E(2) & B1(R2);
          IF TEMP2=B1(R2) THEN DO;
             IF  RR<= V2 THEN DO;
                  R4(RR)=R2+200;
                  RR=RR+1;
             END;
             ELSE DO;
                  GOTO ZZ2;
             END;
          END;
        END;
   RR=RR-1;
   IF (RR¬=V2) & (V2 > 1) THEN GOTO ZZ2;
   IF R4(1)=0 THEN DO;
         DOWN=0;
         HOUR=R1;
         I=BACK(REST(R3));
         CALL PROFSUB;
         IF FREI=1 &FOUND=1 THEN DO;
              IF PRFLAG=1 THEN
              PUT SKIP LIST(CN(COURSE(I)),HOUR,'  NEIGH,FREI');
              IF COURSE(I)<201
                THEN  HB(HOUR,1)=(HB(HOUR,1) | B1(COURSE(I)));
             ELSE HB(HOUR,2)=(HB(HOUR,2) | B1(COURSE(I)-200));
                COURSE(I)=-1*COURSE(I);
                 GOTO ZZ3;
         END;
         GOTO ZZ2;
    END;
```

```
    DO R5=1 TO 4;
        L1=((INDEX(XX,SUBSTR(FA(REST(R3),R5),1,1))-1)*26)
                    +INDEX(XX,SUBSTR(FA(REST(R3),R5),2))-1;
        IF L1<0 THEN GOTO ZZ1;
        IF PTIME (L1,R1)>0 THEN GOTO ZZ2;
    END;
ZZ1:
   DO R9=1 TO RR;
        DO HOUR=1 TO R1-1,R1+1 TO 8;
            NEXTONE=R4(R9);
             OLDONE=R1;
            CALL NEIGH1;
            IF FEIN = 1 THEN GOTO ZZ5;
            IF FEIN = 0 THEN GOTO ZZ4;
            Y8=BACK(R4(R9));
            COURSE(Y8)=ABS(COURSE(Y8));
             HOU=HOUR;
             HO=HOUR;
             IF MOD(HOUR,2) = 0 THEN HO=HOUR-1;
             CALL GUIDE;
            IF FREI=1 & FOUND=1 THEN DO;
                 HOUR=R1;
                 COURSE(Y8)=ABS(COURSE(Y8));
                 IF PRFLAG=1 THEN
                  PUT SKIP LIST
                  (CN(COURSE(Y8)),HOU,'       NEIGH');
                 CALL LOESCH;
                 CALL LOESCHP;
                 IF COURSE(Y8)<201 THEN
                 HB(HOUR,1)=(HB(HOUR,1) & (¬ B1(COURSE(Y8))));
                 ELSE
                HB(HOUR,2)=(HB(HOUR,2) &(¬B1(COURSE(Y8)-200)));
                  HOUR=12;
                  IF R1=1 | R1=2 THEN HOUR=9;
                  IF R1=3 | R1=4 THEN HOUR = 10;
                  IF R1=5 | R1=6 THEN HOUR = 11;
                  CALL LOESCH;
                  CALL LOESCHP;
                 IF COURSE(Y8)<201 THEN
                 HB(HOUR,1)=(HB(HOUR,1) & (¬ B1(COURSE(Y8))));
                 ELSE
                HB(HOUR,2)=(HB(HOUR,2) &(¬B1(COURSE(Y8)-200)));
                  COURSE(Y8)=-1*COURSE(Y8);
             END;
             ELSE DO;
                  COURSE(Y8)=-1*COURSE(Y8);
                  IF HOUR=8 THEN DO;
                       IF R1=8 THEN GOTO ZZ3;
                       GOTO ZZ2;
                  END;
                  IF HOUR=8 THEN GOTO ZZ2;
             END;
ZZ4:
         END;
       END;
ZZ5:
       HOUR = R1;
       I=BACK(REST(R3));
       COURSE(I)=ABS(COURSE(I));
       CALL PROFSUB;
       IF FREI=1 &FOUND=1 THEN DO;
             IF PRFLAG=1 THEN
             PUT SKIP(2) LIST
             (CN(COURSE(I)),HOUR, '         SCHEDULED');
             IF COURSE(I)<201
             THEN  HB(HOUR,1)=(HB(HOUR,1) | B1(COURSE(I)));
```

```
            ELSE HB(HOUR,2)=(HB(HOUR,2) | B1(COURSE(I)-200));
            COURSE(I)=-1*COURSE(I);
            GOTO ZZ3;
        END;
        ELSE DO;
        END;
ZZ2:
END;
ZZ3:
END;
DOWN = 3;
RETURN;
END NEIGH;
/*****************************************************************/
/*    RESCHEDULING OF COURSES, THAT OPPOSE THE RESCHEDULING      */
/*    OF COURSES,WHICH HAVE TO BE RESCHEDULED IN NEIGH           */
/*                                                               */
/*    SUBROUTINE NEIGH1:                                         */
/*                                                               */
/*****************************************************************/
NEIGH1: PROC;
DCL(E(2)                        )BIT (200);
DCL (R4(5),R9,RR,R1,R2,R5,D1,D2)FIXED (3);
D1=HOUR;
D2=Y8;
FEIN=2;
      R1=HOUR;
        RR=1;
        R4=0;
        E(1)=HB(R1,1) & D(BACK(NEXTONE),1);
        E(2)=HB(R1,2) & D(BACK(NEXTONE),2);
        DO R2=1 TO 200;
          TEMP2= E(1) & B1(R2);
          IF TEMP2=B1(R2) THEN DO;
             IF  RR<= 5 THEN DO;
                  R4(RR)=R2;
                  RR=RR+1;
             END;
             ELSE DO;
                  GOTO ZZ22;
             END;
          END;
          TEMP2= E(2) & B1(R2);
          IF TEMP2=B1(R2) THEN DO;
             IF  RR<= 5 THEN DO;
                  R4(RR)=R2+200;
                  RR=RR+1;
             END;
             ELSE DO;
                  GOTO ZZ22;
             END;
          END;
        END;
   RR=RR-1;
   IF R4(1)=0 THEN DO;
             HOUR=R1;
            Y8=BACK(NEXTONE);
            COURSE(Y8)=ABS(COURSE(Y8));
             HOU=HOUR;
             HO=HOUR;
             IF MOD(HOUR,2) = 0 THEN HO=HOUR-1;
             CALL GUIDE;
            IF FREI=1 & FOUND=1 THEN DO;
                 HOUR=D1;
                 COURSE(Y8)=ABS(COURSE(Y8));
                 IF PRFLAG=1 THEN
```

```
                    PUT SKIP LIST
                    (CN(COURSE(Y8)),HOU,'NEIGH1,FREI');
                   CALL LOESCH;
                   CALL LOESCHP;
                   IF COURSE(Y8)<201 THEN
                   HB(HOUR,1)=(HB(HOUR,1) & (¬ B1(COURSE(Y8))));
                   ELSE
                  HB(HOUR,2)=(HB(HOUR,2) &(¬B1(COURSE(Y8)-200)));
                    HOUR=12;
                    IF D1=1 | D1=2 THEN HOUR=9;
                    IF D1=3 | D1=4 THEN HOUR = 10;
                    IF D1=5 | D1=( THEN HOUR = 11;
                    CALL LOESCH;
                    CALL LOESCHP;
                   IF COURSE(Y8)<201 THEN
                   HB(HOUR,1)=(HB(HOUR,1) & (¬ B1(COURSE(Y8))));
                   ELSE
                  HB(HOUR,2)=(HB(HOUR,2) &(¬B1(COURSE(Y8)-200)));
                    COURSE(Y8)=-1*COURSE(Y8);
              FEIN=1;
              GOTO ZZ22;
              END;
         END;
     DO R5=1 TO 4;
         L1=((INDEX(XX,SUBSTR(FA(NEXTONE,R5),1,1))-1)*26)
                  +INDEX(XX,SUBSTR(FA(NEXTONE,R5),2))-1;
         IF L1<0 THEN GOTO ZZ11;
         IF PTIME (L1,R1)>0 THEN GOTO ZZ22;
     END;
ZZ11:
    DO R9=1 TO RR;
         DO HOUR=1 TO R1-1,R1+1 TO 8;
             Y8=BACK(R4(R9));
             COURSE(Y8)=ABS(COURSE(Y8));
              HOU=HOUR;
              HO=HOUR;
              IF MOD(HOUR,2) = 0 THEN HO=HOUR-1;
              CALL GUIDE;
             IF FREI=1 & FOUND=1 THEN DO;
                  HOUR=R1;
                  COURSE(Y8)=ABS(COURSE(Y8));
                  IF PRFLAG=1 THEN
                   PUT SKIP LIST
                   (CN(COURSE(Y8)),HOU,'NEIGH1',R1);
                  I=Y8;
                  CALL LOESCH;
                  CALL LOESCHP;
                  IF COURSE(Y8)<201 THEN
                  HB(HOUR,1)=(HB(HOUR,1) & (¬ B1(COURSE(Y8))));
                  ELSE
                 HB(HOUR,2)=(HB(HOUR,2) &(¬B1(COURSE(Y8)-200)));
                   HOUR=12;
                   IF R1=1 | R1=2 THEN HOUR=9;
                   IF R1=3 | R1=4 THEN HOUR = 10;
                   IF R1=5 | R1=6 THEN HOUR = 11;
                   CALL LOESCH;
                   CALL LOESCHP;
                  IF COURSE(Y8)<201 THEN
                  HB(HOUR,1)=(HB(HOUR,1) & (¬ B1(COURSE(Y8))));
                  ELSE
                 HB(HOUR,2)=(HB(HOUR,2) &(¬B1(COURSE(Y8)-200)));
                   COURSE(Y8)=-1*COURSE(Y8);
    GOTO ZZ44;
              END;
              ELSE DO;
                   COURSE(Y8)=-1*COURSE(Y8);
```

```
                    IF HOUR=8 THEN DO;
                              FEIN=0;
                              GOTO ZZ22;
                    END;
               END;
          END;
ZZ44:
     END;
ZZ22:
HOUR=D1;
Y8=D2;
RETURN;
END NEIGH1;
/*****************************************************************/
/*   INITIATE THE SCHEDULING OF ALL COURSES                      */
/*                                                               */
/*   SUBROUTINE  PARTIAL                                         */
/*                                                               */
/*                                                               */
/*****************************************************************/
PARTIAL: PROC;
   SINGEL=40;
DO HO=1,3,5,7;
 DO UR=0 TO 1;
 HOUR=HO+UR;
 HOU=HOUR;
        DO Y8=1 TO F;
           CALL GUIDE;
        END;
   END;
END;
RETURN;
END PARTIAL;
/*****************************************************************/
/*  SUBROUTINE TO CHECK FOR FREE PROFESSOR                       */
/*                                                               */
/*                                                               */
/*     SUROUTINE  PROFSUB                                        */
/*                                                               */
/*****************************************************************/
PROFSUB:PROC;
DCL (L(4),J1,L1) FIXED(3);
L=-1;
L1=0;
DO J=1 TO 4;
    L(J)=((INDEX(XX,SUBSTR(FA(COURSE(I),J),1,1))-1)*26)
          +INDEX(XX,SUBSTR(FA(COURSE(I),J),2))-1;
    IF L(J)<0 THEN DO;
          J1=J-1;
          GOTO FILL;
      END;
      ELSE DO;
      L1=L1+PTIME(L(J),HOUR);
      END;
END;
J1=4;
FILL:
IF L1¬=0 THEN DO;
     FREI=0;
     RETURN;
END;
/*****************************************************************/
/*    CALCULATE COURSES WITH MORE THAN ONE SEGMENT               */
/*    AND SEVERAL PROFESSORS.                                    */
/*                                                               */
/*                                                               */
/*                                                               */
/*****************************************************************/
```

```
EQ=1;
IF (J1>1)& (COND(COURSE(I))=0) THEN DO;
   DO J=2 TO J1;
        IF L(J-1)¬= L(J) THEN EQ=EQ+1;
   END;
   IF EQ>1 THEN DO;
        NQ=N(COURSE(I));
        N(COURSE(I))=NQ/EQ;
        FREI=1;
        DO J=1 TO EQ;
               KBACK=0;
               CALL ROOMSUB ;
               IF (FOUND ¬=1) THEN DO;
                     N(COURSE(I))=NQ;
                     IF J=1 THEN RETURN;
                     CALL LOESCH;
                     RETURN;
                END;
        END;
        DO J=1 TO J1;
             PTIME(L(J),HOUR)=COURSE(I);
        END;
        N(COURSE(I))=NQ;
        FREI=1;
        RETURN;
    END;
END;
/*****************************************************************/
/*                                                               */
/*                                                               */
/*                                                               */
/*                                                               */
/*                                                               */
/*****************************************************************/
               KBACK=0;
               FREI=1;
               CALL ROOMSUB ;
               IF (FOUND ¬=1) THEN RETURN;
        DO J=1 TO J1;
             PTIME(L(J),HOUR)=COURSE(I);
        END;
  RETURN;
END PROFSUB;
/*****************************************************************/
/*                                                               */
/*   SEMI RANDOM VARIATION  OF THE COURSE ARRAY AND THE          */
/*   COURSE INDICATORS D(*,*)                                    */
/*   SUBROUTINE SWITCH                                           */
/*                                                               */
/*****************************************************************/
SWITCH: PROC;
DCL(M1(400),FRAND(400)      ) FIXED (3);
DCL (M2(400,2)              ) BIT (200);
   M1=0;
   M2='0'B;
   FRAND(1)=1;
   DO I=2 TO F;
        X=RAND(X);
        T1=(X*(I-2))+1;
        FRAND(I)=FRAND(T1);
        FRAND(T1)=I;
   END;
DO I=1 TO F;
   M1(I)=ABS(COURSE(FRAND(I)));
```

```
IF PRFLAG=1 THEN
PUT SKIP LIST (I,FRAND(I),M1(I));
   M2(I,1)=D(FRAND(I),1);
   M2(I,2)=D(FRAND(I),2);
END;
COURSE=M1;
D=M2;
END SWITCH;
/*****************************************************************/
/*                                                               */
/*      SUBROUTINES FOR THE ROOM ASSIGNMENT                      */
/*                                                               */
/*                                                               */
/*                                                               */
/*****************************************************************/
/*****************************************************************/
/*   SUBROUTINE FOR ROOM ASSIGNMENT                              */
/*                                                               */
/*      CHOOSE THE MODE:WITH OR WITHOUT BUILDING BOUNDS          */
/*                      SUBROUTINE ROOMSUB                       */
/*                                                               */
/*****************************************************************/
ROOMSUB: PROC;
IF DOWN=3 THEN GOTO ALLES;
IF DOWN=2 THEN DO;
   CALL ROOMS1;
   RETURN;
END;
        ROOMP=10000;
        ROOME=75000;
        CALL ROOMS1;
        RETURN;
/*****************************************************************/
/*  AE0000 COURSES IN                                            */
/*     HALLIGAN HALL                                             */
/*                                                               */
/*                                                               */
/*                                                               */
/*****************************************************************/
ALLES:
IF SUBSTR(CN(COURSE(I)),1,2) = 'AE'
   THEN DO;
        ROOMP=20000;
        ROOME = 30000;
        CALL ROOMS1;
        RETURN;
   END;
/*****************************************************************/
/*   AS0000, CO0000, CM0000, IS0000, MN0000 COURSES IN           */
/*           INGERSOL HALL (2)  AND THEN                         */
/*           INGERSOL HALL (3)                                   */
/*                                                               */
/*                                                               */
/*****************************************************************/
IF (SUBSTR(CN(COURSE(I)),1,2) = 'AS') |
   (SUBSTR(CN(COURSE(I)),1,2) = 'CO') |
   (SUBSTR(CN(COURSE(I)),1,2) = 'CM') |
   (SUBSTR(CN(COURSE(I)),1,2) = 'IS') |
   (SUBSTR(CN(COURSE(I)),1,2) = 'MN')
   THEN DO;
        ROOMP=42000;
        ROOME = 43000;
        CALL ROOMS1;
        IF FOUND = 1 THEN RETURN;
        ROOMP = 43000;
        ROOME=44000;
        CALL ROOMS1;
        RETURN;
```

```
   END;
/*****************************************************************/
/*      EE0000 COURSES IN                                        */
/*         BULLARD HALL  AND THEN                                */
/*         SPANNAGEL HALL (2,3,4)                                */
/*                                                               */
/*      EW0000                                                   */
/*****************************************************************/
IF (SUBSTR(CN(COURSE(I)),1,2) = 'EE') |
   (SUBSTR(CN(COURSE(I)),1,2) = 'EW')
   THEN DO;
        ROOMP=10000;
        ROOME = 20000;
        CALL ROOMS1;
        IF FOUND = 1 THEN RETURN;
        ROOMP = 72000;
        ROOME=75000;
        CALL ROOMS1;
        RETURN;
   END;
/*****************************************************************/
/*        GH0000 COURSES IN BLD.224                              */
/*                                                               */
/*                                                               */
/*                                                               */
/*                                                               */
/*****************************************************************/
IF SUBSTR(CN(COURSE(I)),1,2) = 'GH'
   THEN DO;
        ROOMP=50000;
        ROOME = 60000;
        CALL ROOMS1;
        RETURN;
   END;
/*****************************************************************/
/*      CS0000, MS0000 COURSES IN SPANNAGEL HALL (2,3,4)         */
/*      SE0000                                                   */
/*                                                               */
/*                                                               */
/*****************************************************************/
IF (SUBSTR(CN(COURSE(I)),1,2) = 'CS') |
   (SUBSTR(CN(COURSE(I)),1,2) = 'MS') |
   (SUBSTR(CN(COURSE(I)),1,2) = 'SE')
   THEN DO;
        ROOMP=72000;
        ROOME = 75000;
        CALL ROOMS1;
        RETURN;
   END;
/*****************************************************************/
/*     ME0000 COURSES IN                                         */
/*         HALLIGAN HALL (2) AND THEN                            */
/*         HALLIGAN HALL (1)                                     */
/*                                                               */
/*****************************************************************/
IF (SUBSTR(CN(COURSE(I)),1,2) = 'ME')
   THEN DO;
        ROOMP=22000;
        ROOME = 30000;
        CALL ROOMS1;
        IF FOUND = 1 THEN RETURN;
        ROOMP = 21000;
        ROOME=22000;
        CALL ROOMS1;
        RETURN;
```

```
   END;
/*****************************************************************/
/*     MA0000 COURSES IN                                         */
/*     INGERSOL HALL (3) AND THEN                                */
/*     INGERSOL HALL (2)                                         */
/*                                                               */
/*                                                               */
/*****************************************************************/
IF (SUBSTR(CN(COURSE(I)),1,2) = 'MA')
   THEN DO;
        ROOMP=43000;
        ROOME = 44000;
        CALL ROOMS1;
        IF FOUND = 1 THEN RETURN;
        ROOMP = 41000;
        ROOME=43000;
        CALL ROOMS1;
        RETURN;
   END;
/*****************************************************************/
/*   OA0000, OS0000 IN                                           */
/*   INGERSOL HALL (3) AND THEN                                  */
/*   ROOT HALL (2) LEFT SIDE                                     */
/*                                                               */
/*                                                               */
/*****************************************************************/
IF (SUBSTR(CN(COURSE(I)),1,2) = 'OA') |
   (SUBSTR(CN(COURSE(I)),1,2) = 'OS')
   THEN DO;
        ROOMP=43000;
        ROOME = 44000;
        CALL ROOMS1;
        IF FOUND = 1 THEN RETURN;
        ROOMP = 62390;
        ROOME=63000;
        CALL ROOMS1;
        RETURN;
   END;
/*****************************************************************/
/*    MR0000 COURSES IN ROOT HALL (2) LEFT SIDE                  */
/*                                                               */
/*                                                               */
/*                                                               */
/*                                                               */
/*****************************************************************/
IF SUBSTR(CN(COURSE(I)),1,2) = 'MR'
   THEN DO;
        ROOMP=62390;
        ROOME = 63000;
        CALL ROOMS1;
        RETURN;
   END;
/*****************************************************************/
/*   NS0000 COURSES IN ROOR HALL MIDDLE PART                     */
/*   ST0000                                                      */
/*                                                               */
/*                                                               */
/*                                                               */
/*****************************************************************/
IF( SUBSTR(CN(COURSE(I)),1,2) = 'NS') |
   (SUBSTR(CN(COURSE(I)),1,2) = 'ST')
   THEN DO;
        ROOMP=60000;
        ROOME = 62390;
        CALL ROOMS1;
        RETURN;
```

```
   END;
/*****************************************************************/
/*   OC0000 COURSES IN ROOT HALL RIGHT SIDE                      */
/*                                                               */
/*                                                               */
/*                                                               */
/*                                                               */
/*****************************************************************/
IF SUBSTR(CN(COURSE(I)),1,2) = 'OC'
   THEN DO;
        ROOMP=60000;
        ROOME = 62390;
        CALL ROOMS1;
        RETURN;
   END;
/*****************************************************************/
/*   PH0000 COURSES IN SPANNAGEL HALL (1)                        */
/*   CC0000    AND THEN  SPANNAGEL HALL (2)                      */
/*                                                               */
/*                                                               */
/*                                                               */
/*****************************************************************/
IF SUBSTR(CN(COURSE(I)),1,2) = 'PH' |
   (SUBSTR(CN(COURSE(I)),1,2) = 'CC')
   THEN DO;
        ROOMP=70000;
        ROOME = 72000;
        CALL ROOMS1;
        IF FOUND = 1 THEN RETURN;
        ROOMP = 72000;
        ROOME=73000;
        CALL ROOMS1;
        RETURN;
   END;
END ROOMSUB;
/*****************************************************************/
/*   SUBROUTINE  TO LOOK FOR A POSSIBLE FREE ROOM                */
/*                                                               */
/*                                                               */
/*                                                               */
/*                                                               */
/*****************************************************************/
ROOMS1: PROC;
DCL (J,SUM                                        )FIXED;
FOUND=0;
J=1;
DO WHILE (ROOMN(J)<ROOME);
   IF ROOMN(J)<ROOMP THEN GOTO WEITER2;
      IF TIMER(J,HOUR)¬=0 THEN GOTO WEITER2;
      IF  HIGH*N(COURSE(I))<PLACES(J)&PLACES(J)>SINGEL
                                    THEN GOTO WEITER2;
      IF LOW*N(COURSE(I))<PLACES(J) THEN DO;
         TIMER(J,HOUR)=COURSE(I);
         FOUND=1;
         RETURN;
      END;
        ELSE DO;
             SUM=PLACES(J);
             KBACK=J;
             CALL ROOMS2 (J+1,SUM);
             IF FOUND=1 THEN DO;
                             TIMER(J,HOUR)=COURSE(I);
                  RETURN;
             END;
```

```
              ELSE DO;
IF FOUND > 1 THEN J=KBACK-1;
              END;
         END;
WEITER2:
J=J+1;
END;
FOUND=0;
RETURN;
END ROOMS1;
/*****************************************************************/
/*   SUROUTINE TO LOOK FOR A POSSIBLE SECOND ROOM                */
/*                                                               */
/*                                                               */
/*                                                               */
/*                                                               */
/*****************************************************************/
ROOMS2:PROC (K,SUM);
DCL (J,K,SUM                                          )FIXED;
DO J=K TO R WHILE (ROOMN(J)<=ROOME);
         FIRST=ROOMN(K-1)/1000;
         SECOND=ROOMN(J)/1000;
         IF SECOND-FIRST> 0 THEN DO;
              FOUND=2;
              KBACK=J;
              RETURN;
         END;
       IF TIMER(J,HOUR)¬=0 THEN GOTO WEITER3;
       IF  HIGH*N(COURSE(I))<(PLACES(J)+SUM) &PLACES(J)>SINGEL
                                              THEN GOTO WEITER3;
       IF LOW*N(COURSE(I))<PLACES(J)+SUM THEN DO;
          TIMER(J,HOUR)=COURSE(I);
          FOUND=1;
          RETURN;
       END;
         ELSE DO;
              SUM=SUM+PLACES(J);
              CALL ROOMS3 (J+1,SUM);
              IF FOUND=1 THEN DO;
                              TIMER(J,HOUR)=COURSE(I);
                        RETURN;
              END;
              ELSE DO;
IF FOUND>1 THEN RETURN;
              SUM=SUM-PLACES(J);
              END;
         END;
WEITER3:
END;
FOUND=0;
RETURN;
END ROOMS2;
/*****************************************************************/
/*   SOUBROUTINE TO LOOK FOR A POSSIBLE THIRD  ROOM              */
/*                                                               */
/*                                                               */
/*                                                               */
/*                                                               */
/*****************************************************************/
ROOMS3:PROC (K,SUM);
DCL (J,K,SUM                                          )FIXED;
DO J=K TO R WHILE (ROOMN(J)<=ROOME);
/*       IF ABS(ROOMN(K-1)-ROOMN(J))/1000> 1 THEN DO*/
         FIRST=ROOMN(K-1)/1000;
         SECOND=ROOMN(J)/1000;
```

```
        IF SECOND-FIRST> 0 THEN DO;
             FOUND=3;
             KBACK=J;
             RETURN;
        END;
      IF TIMER(J,HOUR)¬=0 THEN GOTO WEITER4;
   IF (1.2*HIGH*N(COURSE(I))<(PLACES(J)+SUM))
        &(PLACES(J)>SINGEL)  THEN GOTO WEITER4;
      IF 0.95*LOW*N(COURSE(I))<PLACES(J)+SUM THEN DO;
         TIMER(J,HOUR)=COURSE(I);
         FOUND=1;
         RETURN;
      END;
        ELSE DO;
             SUM=SUM+PLACES(J);
             CALL ROOMS4 (J+1,SUM);
             IF FOUND=1 THEN DO;
                        TIMER(J,HOUR)=COURSE(I);
                  RETURN;
             END;
             ELSE DO;
IF FOUND>1 THEN RETURN;
             SUM=SUM-PLACES(J);
             END;
        END;
WEITER4:
END;
FOUND=0;
RETURN;
END ROOMS3;
/*****************************************************************/
/*  SUBROUTINE TO LOOK FOR A POSSIBLE FOURTH ROOM                */
/*                                                               */
/*                                                               */
/*                                                               */
/*                                                               */
/*****************************************************************/
ROOMS4:PROC (K,SUM);
DCL (J,K,SUM                                        )FIXED;
DO J=K TO R WHILE (ROOMN(J)<=ROOME);
    /*  IF ABS(ROOMN(K-1)-ROOMN(J))/1000> 1 THEN DO*/
        FIRST=ROOMN(K-1)/1000;
        SECOND=ROOMN(J)/1000;
        IF SECOND-FIRST> 0 THEN DO;
             FOUND=4;
             KBACK=J;
             RETURN;
        END;
      IF TIMER(J,HOUR)¬=0 THEN GOTO WEITER5;
      IF 1.5* HIGH*N(COURSE(I))<(PLACES(J)+SUM)
          &PLACES(J)>SINGEL  THEN GOTO WEITER5;
      IF 0.9*LOW*N(COURSE(I))<PLACES(J)+SUM  THEN DO;
         TIMER(J,HOUR)=COURSE(I);
         FOUND=1;
         RETURN;
        END;
WEITER5:
END;
FOUND=0;
RETURN;
END ROOMS4;
/*****************************************************************/
/*   END OF SUBROUTINES                                          */
/*                                                               */
/*                                                               */
/*                                                               */
/*****************************************************************/
END FINAL;
```

# APPENDIX B

## FINAL PLC OUTPUT: ROOM SCHEDULE (EXAMPLE PAGE)

ROOM SCHEDULE FOR FINALS

| ROOM | MO 1 | MO 2 | TU 1 | TU 2 | WE 1 | WE 2 | TH 1 | TH 2 | MO 3 | TU 3 | WE 3 | TH 3 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| B104 | EE4485 | EE2212 | EE2812 | EE3310 | EE2401 | EE2810 | EE2213 | ------ | EE4481 | EE4565 | EE3714 | EE2101 |
| B125 | ------ | EE2212 | EE2812 | EE3310 | EE2401 | EE2810 | EE4415 | EE4910 | ------ | EE4416 | EE3714 | EE2101 |
| B202 | ------ | EE3118 | ------ | EE2411 | ------ | ------ | ------ | ------ | ------ | EE2418 | ------ | ------ |
| H109 | AE2021 | AE4101 | ME2501 | ME3220 | ------ | AE4702 | AE4706 | AE2801 | ------ | ------ | AE4304 | ------ |
| H121A | AE2021 | ------ | ME2201 | ME3220 | AE2042 | AE4501 | ME2440 | ME4162 | ------ | ------ | ------ | ------ |
| H121B | AE4307 | ------ | ME2201 | AE4323 | ------ | AE2036 | AE4712 | AE2801 | ------ | ------ | ------ | ------ |
| H123 | AE3005 | ------ | AE4305 | ------ | ------ | ------ | ------ | ------ | ------ | ------ | ------ | ------ |
| H125 | AE4342 | ------ | AE2015 | ------ | ------ | ------ | ------ | AE4452 | ------ | ------ | ------ | ------ |
| H201E | ME4240 | ------ | ME3230 | ------ | ME4525 | ME2101 | ME3711 | ME3611 | ME2410 | ------ | ------ | ------ |
| H201F | ME4801 | ------ | ME3230 | ME4613 | ------ | ME2101 | ME3711 | ME3611 | ------ | ------ | ------ | ------ |
| I119 | ------ | ------ | ------ | ------ | ------ | ------ | ------ | ------ | MA3400 | ------ | ------ | ------ |
| I249 | MN4154 | MA2047 | IS4185 | MN4145 | MN4761 | MN4163 | MN3301 | IS3183 | IS2000 | AS1501 | MN3305 | ------ |
| I260 | MN4154 | MA2047 | MN3140 | MN4145 | ------ | ------ | IS4183 | IS3183 | IS2000 | ------ | ------ | ------ |
| I263 | MN4154 | MA2047 | IS4185 | MN4145 | MN3371 | ------ | ------ | IS3183 | MN4122 | ------ | MN3305 | ------ |
| I265 | MN4371 | MA2047 | MN4116 | MN4145 | MN2155 | MN3105 | ------ | MA1112 | ------ | ------ | MN4123 | ------ |
| I267 | MN3161 | MA2047 | MN3140 | CO3112 | MN2155 | MN3105 | ------ | ------ | ------ | ------ | ------ | ------ |
| I271 | MN3161 | IS3171 | ------ | ------ | ------ | MN3105 | ------ | ------ | ------ | ------ | ------ | ------ |
| I280 | MN4159 | IS3171 | ------ | CO3112 | MN2155 | MN3105 | ------ | ------ | ------ | ------ | ------ | ------ |
| I282 | MN3001 | ------ | ------ | ------ | ------ | MN3105 | ------ | ------ | ------ | ------ | ------ | ------ |
| I285 | MN4110 | ------ | ------ | ------ | ------ | MN3105 | ------ | ------ | ------ | ------ | ------ | ------ |
| I323 | MA2181 | ------ | OA3501 | MA3132 | OS3104 | OA3301 | OA4702 | OS3604 | OA3401 | MA1118 | ------ | MA1116 |
| I325 | MA4393 | ------ | MA2129 | MA3132 | OA4603 | OA3301 | OA4702 | OS3604 | OA3401 | OA4202 | ------ | MA1116 |
| I361 | MA1117 | ------ | MA3605 | MA3132 | OA4603 | OA3301 | OS3105 | ------ | ------ | ------ | ------ | MA1116 |
| I365 | OA4704 | ------ | OS4701 | MA2121 | ------ | ------ | OS3105 | OS3636 | ------ | ------ | ------ | ------ |
| I366 | OA2600 | OS3006 | OS3303 | MA2121 | ------ | ------ | ------ | OS3636 | OA3601 | ------ | ------ | ------ |
| I368 | OS3005 | OS3006 | MA2042 | MA2121 | ------ | ------ | ------ | MA3232 | OA3601 | ------ | ------ | ------ |
| I369 | OA2600 | OS3006 | MA2042 | MA2121 | ------ | ------ | ------ | MA3232 | ------ | ------ | ------ | ------ |
| I379 | OS3004 | OS3006 | OS3002 | OA4206 | ------ | ------ | ------ | OA4306 | MA1115 | ------ | ------ | ------ |
| I381 | OS3004 | OS3006 | OS3602 | OS3404 | ------ | ------ | ------ | MA2025 | OA3101 | ------ | ------ | ------ |
| I386 | ------ | OS3006 | ------ | ------ | ------ | ------ | ------ | MA2025 | OA3101 | ------ | ------ | ------ |
| I387 | ------ | ------ | ------ | ------ | ------ | ------ | ------ | OS3401 | MA3362 | ------ | ------ | ------ |
| O105B | ------ | GH3902 | ------ | ------ | GH4907 | ------ | ------ | ------ | ------ | ------ | ------ | ------ |
| O112 | ------ | GH3902 | ------ | ------ | ------ | ------ | ------ | ------ | ------ | ------ | ------ | ------ |
| R200A | OC3240 | OC3261 | OC3120 | OC3212 | NS3030 | NS3153 | NS3662 | OC4213 | NS4690 | NS3720 | OC2120 | ------ |
| R200C | NS3040 | ------ | NS3631 | OC3212 | NS3030 | NS3153 | ST1810 | ------ | NS4310 | NS4251 | OC2120 | ------ |
| R200D | NS3040 | NS2154 | NS3280 | NS3021 | NS4010 | NS3153 | ST1810 | ------ | OC4212 | NS4251 | ------ | ------ |
| R200E | NS3452 | ------ | NS3840 | NS3021 | NS4010 | NS3153 | NS4420 | ------ | ------ | ------ | ------ | ------ |
| R204 | NS4830 | ------ | NS3361 | ------ | ------ | ------ | NS4420 | ------ | ------ | ------ | ------ | ------ |
| R204A | NS4300 | NS4261 | ------ | ------ | NS4710 | ------ | NS3320 | NS3620 | ------ | ------ | ------ | ------ |
| R208 | OC2001 | ------ | ------ | ------ | ------ | NS3665 | OC3230 | ------ | ------ | ------ | ------ | ------ |
| R210 | NS4410 | ------ | ------ | NS3701 | ------ | ------ | NS3410 | ------ | ------ | ------ | ------ | ------ |

# APPENDIX C

## FINAL PLC OUTPUT: FACULTY SCHEDULE (EXAMPLE PAGE)

FACULTY SCHEDULE FOR FINALS

| PROF | MO 1 | MO 2 | TU 1 | TU 2 | WE 1 | WE 2 | TH 1 | TH 2 | MO 3 | TU 3 | WE 3 | TH 3 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| AA | | | | | | | | | | | | |
| AB | | | | | | | | | | | | |
| AC | | | | | | | | | | | | |
| AD | | | | | | | | | | | | |
| AE | | | | | | | | | | | | |
| AF | | | | | | | | | | | | |
| AG | | | | | EE4550 | | | | | | | |
| AH | | | | | | | | | | | | |
| AI | | | | | | | | | | | | |
| AJ | | | | | | | | | | | | |
| AK | | | | | | | | | | | | |
| AL | | | | | | | | | | | | |
| AM | | | | | | | | | | | | |
| AN | | | | | | | | | | | | |
| AO | | | | | | | | | | | | |
| AP | | | | | | | | | | | | |
| AQ | | | | | | | | | | | | |
| AR | | | | PH3321 | | | | | | PH3190 | | |
| AS | | | | | | | | | | | | |
| AT | EE3800 | | | | | | | | | | | |
| AU | | | | | | | | | | | | |
| AV | | | | | | | | | | | | |
| AW | | | | | | | | | | | | |
| AX | | | | | | | | | | | | |
| AY | | | | | | | | | | | | |
| AZ | | | | | | | | | | | | |
| BA | | | | | | | | | | | | |
| BB | | | EE2422 | | | | | | | | | |
| BC | | | | | | | | | | | | |
| BD | | IS3171 | | | | | | | | | | |
| BE | | | AE2015 | | | AE2036 | | | | | | |
| BF | | OC3261 | | OC3212 | | | | | | | | |
| BG | | | | | | | | | | | | |
| BH | | | | | | | | | | | | |
| BI | | | | | | AE4501 | | | | | | |
| BJ | | | | | | | | | | | | |
| BK | | | | | | | | | | | | |
| BL | | | | | | | | | | | | |
| BM | | | | | | | | | | | | |
| BN | | | | | | | | | | | | |
| BO | | | | | | | | | | | | |

# APPENDIX D

## PRIOR PLC: INPUT PREPARATION PROGAM FOR FINAL PLC

```
//DIETMAR JOB (0102,0034),'ANYTHING',CLASS=G
//*MAIN ORG=NPGVM1.1307P
//  EXEC PLIXCLG
//SYSIN  DD  *
 BUILTN: PROC OPTIONS (MAIN);
 /*****************************************************************/
 /*                                                               */
 /*    THESIS    ROUTINE TO BUILT A NETWORK OF COURSES          */
 /*    AUTHOR:                F I E G A S                        */
 /*    INSTRUCTOR:            K. WOOD                            */
 /*    DATE:               08 APR 1985                           */
 /*                                                               */
 /*****************************************************************/
 /*****************************************************************/
 /*    ATENTION:                                                  */
 /*              CROSSL IS ONE COURSE SHORT(THE LAST ONE)         */
 /*    AND                                                        */
 /*        IF THERE IS A COURSE WITH SECTIONS TO NO OTHER NODE    */
 /*             IT IS NOT USED (HOLE IN NUMBERING)                */
 /*****************************************************************/
 /*****************************************************************/
 /*  N...# OF SECTIONS; C(N)...# OF COURSES                       */
 /*  I,L,K...COUNTERS,  MEMORY,NOW...INTERMEDIAT STORAGE          */
 /*                                                               */
 /*                                                               */
 /*                                                               */
 /*****************************************************************/
 DCL
 1 MATRIX (6000),
  2 INPUT,
    3 COURSE CHAR (6),/* COLUMN OF COURSE NUMBERS */
    3 SECT CHAR (4)  , /* COLUMN OF SECTION NUMBERS */
    3 SN FIXED,
    3 NUMBER FIXED, /*COLUMN NUMBER OF STUDENTS IN SECTION */
  2 C FIXED; /*INTEGER NUMBER CORRESPONDING TO COURSE NUMBER */
 DCL (FLAG,N,L,I,K                          ) FIXED INIT (0);
 DCL (NOW) CHAR(6);
 DCL (D(400,2),B1(200)                      ) BIT  (200);
 DCL (HB                                    ) BIT (200) VAR;
 DCL CROSSL FILE STREAM;
 DCL PRIOR FILE STREAM;
 DCL YBN2 FILE STREAM;
 DCL 1 TRANS (400),
        /*TRANSLATES LIST OF COURSES WITH # OF STUDENTS*/
        2 CN CHAR (6),
        2 TOTAL FIXED;
 OPEN FILE (CROSSL) OUTPUT;
 OPEN FILE (PRIOR) OUTPUT;
 OPEN FILE (YBN2) INPUT;
 /*****************************************************************/
 /*                                                               */
 /*                                                               */
 /*                                                               */
 /*                                                               */
 /*                                                               */
 /*****************************************************************/
```

```
/**/
/*   SET ENDFILE CONDITION*/
/**/
FLAG=0;
ON ENDFILE (YBN2) FLAG=1;
/*****************************************************************/
/*                                                               */
/*                                                               */
/*                                                               */
/*                                                               */
/*                                                               */
/*****************************************************************/
/**/
/*     READ IN DATA      */
/**/
K=1;
NUMBER=0;
C=0;
C(1)=1;
B1(1)='1';
D='0'B;
HB='0'B;
AGAIN:
GET FILE (YBN2) EDIT
(COURSE(1),SECT(1),SN(1),NUMBER(1))(SKIP,A(6),A(4),F(2),F(2));
CN(1)=COURSE(1);
TOTAL(1)=NUMBER(1);
IF INDEX(COURSE(1),'0810')=3 THEN GO TO AGAIN;
DO I=2 TO 4000;
DISCARD:
     GET FILE (YBN2) EDIT
     (COURSE(I),SECT(I),SN(I),NUMBER(I))
               (SKIP,A(6),A(4),F(2),F(2));
     IF (FLAG=1) THEN GO TO SCHLUSS;
/*****************************************************************/
/*                                                               */
/*       EVERY QUARTER                                           */
/*                                                               */
/*                                                               */
/*                                                               */
/*****************************************************************/
     IF INDEX(COURSE(I),'0810')=3 THEN GO TO DISCARD;
     IF INDEX(COURSE(I),'0001')=3 THEN GO TO DISCARD;
/*****************************************************************/
/*                                                               */
/*                                                               */
/*      ONLY THIS QUARTER                                        */
/*                                                               */
/*                                                               */
/*****************************************************************/
     IF INDEX(COURSE(I),'EW0002')=1 THEN GO TO DISCARD;
     IF INDEX(COURSE(I),'CS3800')=1 THEN GO TO DISCARD;
     IF INDEX(COURSE(I),'CS3900')=1 THEN GO TO DISCARD;
     IF INDEX(COURSE(I),'CS4910')=1 THEN GO TO DISCARD;
     IF INDEX(COURSE(I),'EE4900')=1 THEN GO TO DISCARD;
     IF INDEX(COURSE(I),'FL')=1 THEN GO TO DISCARD;
     IF INDEX(COURSE(I),'NS0030')=1 THEN GO TO DISCARD;
     IF INDEX(COURSE(I),'NS3279')=1 THEN GO TO DISCARD;
     IF INDEX(COURSE(I),'NS3379')=1 THEN GO TO DISCARD;
     IF INDEX(COURSE(I),'NS3772')=1 THEN GO TO DISCARD;
     IF INDEX(COURSE(I),'NS3773')=1 THEN GO TO DISCARD;
     IF INDEX(COURSE(I),'NS3775')=1 THEN GO TO DISCARD;
     IF INDEX(COURSE(I),'NS3779')=1 THEN GO TO DISCARD;
     IF INDEX(COURSE(I),'NS4230')=1 THEN GO TO DISCARD;
     IF INDEX(COURSE(I),'NS4671')=1 THEN GO TO DISCARD;
```

```
     IF INDEX(COURSE(I),'NS4679')=1 THEN GO TO DISCARD;
     IF INDEX(COURSE(I),'NS4771')=1 THEN GO TO DISCARD;
     IF INDEX(COURSE(I),'NS4776')=1 THEN GO TO DISCARD;
     IF INDEX(COURSE(I),'ME4902')=1 THEN GO TO DISCARD;
     IF INDEX(COURSE(I),'MR4900')=1 THEN GO TO DISCARD;
     IF INDEX(COURSE(I),'PH1012')=1 THEN GO TO DISCARD;
     IF INDEX(COURSE(I),'PH4998')=1 THEN GO TO DISCARD;
     IF INDEX(COURSE(I),'MN4652')=1 THEN GO TO DISCARD;
     IF INDEX(COURSE(I),'MA1110')=1 THEN GO TO DISCARD;
     IF INDEX(COURSE(I),'MA3002')=1 THEN GO TO DISCARD;
     IF INDEX(COURSE(I),'MA3237')=1 THEN GO TO DISCARD;
     IF INDEX(COURSE(I),'MA4593')=1 THEN GO TO DISCARD;
/*****************************************************************/
/*                                                               */
/*                                                               */
/*                                                               */
/*       END OF TAKING OUT COURSES                               */
/*                                                               */
/*****************************************************************/
     IF INDEX (COURSE(I),COURSE(I-1))=1 THEN DO;
         TOTAL(K)=TOTAL(K)+NUMBER(I);
         C(I)=K;
       END;
     ELSE DO;
         K=K+1;
         TOTAL(K)=NUMBER(I);
         CN(K)=COURSE(I);
         C(I)=K;
       PUT FILE (CROSSL) EDIT (K-1,CN(K-1),TOTAL(K-1))
         (SKIP,F(3),X(5),A(6),X(5),F(3));
    END;
END;
SCHLUSS:      N=I-1;
PUT FILE (CROSSL) EDIT (K,CN(K),TOTAL(K))
         (SKIP,F(3),X(5),A(6),X(5),F(3));
CLOSE FILE (CROSSL);
CLOSE FILE (YBN2);
/*****************************************************************/
/*   END OF INTEGER-COURSE-NUMBER ROUTINE                        */
/*                                                               */
/*                                                               */
/*                                                               */
/*                                                               */
/*****************************************************************/
DO I=2 TO 200;
   B1(I)=(HB || '1'B);
   HB=HB || '0'B;
END;

/*****************************************************************/
/*                                                               */
/*                                                               */
/*                                                               */
/*                                                               */
/*                                                               */
/*****************************************************************/
K=1;
DO I=1 TO N;
   DO J=1 TO I-1, I+1 TO N;
        IF ((INDEX(SECT(I),SECT(J))=1) &
            (SN(I)=SN(J))) THEN DO;
             I1=1;
             IF C(J)>200 THEN I1=2;
             D(K,I1)=D(K,I1) | B1(C(J)-((I1-1)*200));
        END;
   END;
   IF C(I)¬=C(I+1) THEN K=K+1;
END;
```

```
 DO I=1 TO K-1;
    PUT FILE (PRIOR) EDIT (I,D(I,1),D(I,2),TOTAL (I))
                  (SKIP, F(3),X(2),B(200),B(200),X(5),F(3));
 END;
 CLOSE FILE (PRIOR);
 END BUILTN;
//GO.PRIOR DD DSN=S1307.PRIOR,DISP=(SHR,KEEP),
//   VOL=SER=MVS004,UNIT=3350,
//   DCB=(RECFM=FB,LRECL=500,BLKSIZE=5000),SPACE=(CYL,(1,1))
//GO.CROSSL DD DSN=S1307.CROSSL,DISP=(SHR,KEEP),
//   VOL=SER=MVS004,UNIT=3350,
//   DCB=(RECFM=FB,LRECL=80,BLKSIZE=80),SPACE=(CYL,(1,1))
//GO.YBN2   DD  *
AE0810AC3401 1
AE0810AC4101 1
AE0810AC4102 1
AE0810AC4103 1
AE0810AC4201 1
AE0810AC4202 2
```

# APPENDIX E
# QUESTIONNAIRE SCREEN

```
FACULTY NAME:  SMITH     MAILCODE:   SM   DEPT NUMBER:      39
COURSE :  AX2810  YEAR AND QUARTER:   854     SEGMENT:       1

ACCELERATED/WEEK1-6  (1) OR REFRESHER/WEEK7-12 COURSE..?    0
DO YOU NEED SPECIAL STUDENTS IN ONE SEGMENT.........?       N
ARE THERE OTHER FACULTY TEACHING SOME SEGMENT........?      N
I TEACH THIS COURSE IN TEAM WITH ANOTHER FACULTY....?       N

*0...DON'T CARE  1...LIKE TO HAVE *  \SPECIAL HOURS?:       200000022
*2...DON'T LIKE  3...NEED TO HAVE *  /SPECIAL  DAYS?:       111020
THE FINAL IS SCHEDULED..:    Y  SPLIT LAB IN EQUAL SEGM.:    1
DO NOT SCHEDULE AT SAME TIME WITH COURSE (NUMBER)...:       0A4109
DO SCHEDULE TOGETHER WITH COURSE (NUMBER)...:               XX9999
SCHEDULE BACK TO BACK (1), NOT BACK TO BACK (2)...:         0
I TEACH IN SPECIAL BLOCKS:  LECTURE:     1   LAB:           2
NEED MORE/LESS HOURS THAN IN CATALOG: LEC:  N    LAB:      N
need a special room  lect, 1st choice:  X9999  2nd choice: X9999
                     lab , 1st choice:  1364E  2nd choice: 1119
there are other constraints, comments:  0 COMMENT


next course  AX3190     segment   2
```

# APPENDIX F

## CON FOCEXEC: THE CONSTRAINT QUESTIONNAIRE EXEC

```
MODIFY FILE CON
COMPUTE
DATEF=TODAY (DATEF);
TIMEF= HHMMSS (TIMEF);
NXTCRS/A6=;
NXTSG/A1=;
XNAME/A16=;
CRTFORM LINE 1
"ENTER PROF NAME <XNAME  "
"HIT ENTER TO CONTINUE "
COMPUTE
PNAME=XNAME;
MATCH PNAME
ON NOMATCH GOTO ADDPRF
ON MATCH CRTFORM LINE 1
"CURRENT DATA FOR PROFESSOR <D.PNAME> MAILCODE : <T.MC> DEP NUMB:<T.DP>"
"ENTER COURSE:<NXTCRS>  SEGMENT:<NXTSG> AND YEAR/QUARTER:<YQ>"
"HIT ENTER TO CONTINUE"
ON MATCH UPDATE MC DP YQ
ON MATCH IF NXTCRS EQ '      ' THEN GOTO TYPEM;
ON MATCH IF NXTCRS EQ '?     ' OR NXTSG EQ '?' THEN GOTO HELP1;
ON MATCH IF NXTSG EQ ' ' THEN GOTO TYPEM;
ON MATCH GOTO UPDCRS
*-------------------------------------------------------
CASE TYPEM
TYPE "YOU FORGOT TO ENTER COURSE OR SEGMENT"
ENDCASE
*-------------------------------------------------------------------------
*-------------------------------------------------------
CASE HELP1
TYPE "COURSE HAS TO LOOK LIKE 'AX2810'"
TYPE "SEGMENT HAS TO LOOK LIKE '1' IF THERE IS ONLY ONE SEGMENT"
TYPE "        OR IT IS THE FIRST,'2' IF IT IS THE SECOND ...."
ENDCASE
*-------------------------------------------------------------------------
CASE UPDCRS
TYPE "ENTRING CASE UPDCRS"
COMPUTE
PNAME=PNAME;
MC=MC;
CN=NXTCRS;
SG=NXTSG;
MATCH CN SG
ON NOMATCH GOTO ADDCRS
ON MATCH COMPUTE
NXTCRS='XXXXXX';
ON MATCH CRTFORM LINE 1
"CURRENT DATA FOR PROFESSOR <D.PNAME> MAILCODE : <D.MC> DEP NUMB:<D.DP>"
"COURSE <D.CN> FOR YEAR AND QTR:<D.YQ> AND SEGMENT:<55<D.SG>IN FILE"
"ACCELERATED/WEEK1-6  (1) OR REFRESHER/WEEK7-12 COURSE..?<T.ACCEL>"
"DO YOU NEED SPECIAL STUDENTS IN ONE SEGMENT.....? <55 <T.SPECSTUD>"
"ARE THERE OTHER FACULTY TEACHING SOME SEGMENT.....? <55 <T.OTHPROF>"
"I TEACH THIS COURSE IN TEAM WITH ANOTHER FACULTY....?<55 <T.TEAM>"
```

```
"*0...DON'T CARE  1...LIKE TO HAVE *  \SPECIAL HOURS?: <T.SPHOURS>"
"*2...DON'T LIKE  3...NEED TO HAVE *  /SPECIAL  DAYS?: <T.SPDAY>"
"THE FINAL IS SCHEDULED..:< T.FINAL> SPLIT LAB IN EQUAL SEGM.:<T.SPLIT>"
"DO NOT SCHEDULE AT SAME TIME WITH COURSE (NUMBER)...:<55 <T.OTHCOURSE>"
"DO SCHEDULE TOGETHER WITH COURSE (NUMBER)...:<55 <T.TOGCOURSE>"
"SCHEDULE BACK TO BACK (1), NOT BACK TO BACK (2)...:<55 <T.BACK>"
"I TEACH IN SPECIAL BLOCKS:  LECTURE<47 <T.SPBLOLEC> LAB<55 <T.SPBLOLAB>"
"I NEED MORE/LESS HOURS THAN IN CATALOG: LEC:<47<T.MHRSLEC>LAB:<T.MHRSLAB>"
"I NEED A SPECIAL ROOM  LECT, 1ST CHOICE:<T.RMLEC1>  2ND CHOICE:<T.RMLEC2>"
"                       LAB , 1ST CHOICE:<T.RMLAB1>  2ND CHOICE:<T.RMLAB2>"
"THERE ARE OTHER CONSTRAINTS, COMMENTS:  <T.COMMENT1>"
"<T.COMMENT2>"
"<T.COMMENT3>"
"NEXT COURSE <T.NXTCRS  SEGMENT <NXTSG> "
 ON MATCH UPDATE  SPNAME
 ON MATCH UPDATE  MAILC
 ON MATCH UPDATE  DEPT
 ON MATCH UPDATE  YRQTR
 ON MATCH UPDATE  DATEF
 ON MATCH UPDATE  TIMEF
 ON MATCH UPDATE  SPECSTUD
 ON MATCH UPDATE  TOGCOURSE
 ON MATCH UPDATE  ACCEL
 ON MATCH UPDATE  SPLIT
 ON MATCH UPDATE  BACK
 ON MATCH UPDATE  OTHPROF
 ON MATCH UPDATE  TEAM
 ON MATCH UPDATE  SPHOURS
 ON MATCH UPDATE  SPDAY
 ON MATCH UPDATE  SPBLOLEC
 ON MATCH UPDATE  SPBLOLAB
 ON MATCH UPDATE  FINAL
 ON MATCH UPDATE  OTHCOURSE
 ON MATCH UPDATE  RMLEC1
 ON MATCH UPDATE  RMLEC2
 ON MATCH UPDATE  RMLAB1
 ON MATCH UPDATE  RMLAB2
 ON MATCH UPDATE  MHRSLEC
 ON MATCH UPDATE  MHRSLAB
 ON MATCH UPDATE  COMMENT1
 ON MATCH UPDATE  COMMENT2
 ON MATCH UPDATE  COMMENT3
ON MATCH IF NXTCRS='XXXXXX' THEN GOTO TOP;
ON MATCH GOTO UPDCRS
ENDCASE
*-------------------------------------------------------------------------
CASE ADDCRS
TYPE "ENTERING CASE ADDCRS "
COMPUTE
    YQ='813';
    COURSE='XX9999';
    DP='99';
    SEGMENT='1';
    SPECSTUD='N';
    OTHPROF='N';
    TEAM='N';
    SPHOURS='000000000';
    SPDAY='000000';
    SPBLOLEC='1';
    SPBLOLAB='1';
    FINAL='Y';
    OTHCOURSE='XX9999';
    TOGCOURSE='XX9999';
    ACCEL='0';
```

```
    SPLIT='1';
    BACK='0';
    RMLEC1='X9990';
    RMLEC2='X9990';
    RMLAB1='X9990';
    RMLAB2='X9990';
    MHRSLEC='N';
    MHRSLAB='N';
    COMMENT1='0 COMMENT';
PNAME=PNAME;
CN=NXTCRS;
SG=NXTSG;
NXTCRS='YYYYYY';
MATCH CN SG
ON MATCH REJECT
ON NOMATCH CRTFORM LINE 2
"ADDING DATA COURSE: <T.CN> YEAR AND QUARTER: <T.YQ>,SEGMENT:<T.SG>"
"ACCELERATED/WEEK1-6  (1) OR REFRESHER/WEEK7-12 COURSE..?<T.ACCEL>"
"DO YOU NEED SPECIAL STUDENTS IN ONE SEGMENT.....? <55 <T.SPECSTUD>"
"ARE THERE OTHER FACULTY TEACHING SOME SEGMENT.....? <55 <T.OTHPROF>"
"I TEACH THIS COURSE IN TEAM WITH ANOTHER FACULTY....?<55 <T.TEAM>"
"*0...DON'T CARE  1...LIKE TO HAVE *  \SPECIAL HOURS?: <T.SPHOURS>"
"*2...DON'T LIKE  3...NEED TO HAVE *  /SPECIAL  DAYS?: <T.SPDAY>"
"THE FINAL IS SCHEDULED..:< T.FINAL> SPLIT LAB IN EQUAL SEGM.:<T.SPLIT>"
"DO NOT SCHEDULE AT SAME TIME WITH COURSE (NUMBER)...:<55 <T.OTHCOURSE>"
"DO SCHEDULE TOGETHER WITH COURSE (NUMBER)...:<55 <T.TOGCOURSE>"
"SCHEDULE BACK TO BACK (1), NOT BACK TO BACK (2)...:<55 <T.BACK>"
"I TEACH IN SPECIAL BLOCKS:  LECTURE <47<T.SPBLOLEC> LAB<55 <T.SPBLOLAB>"
"I NEED MORE/LESS HOURS THAN IN CATALOG: LEC:<47<T.MHRSLEC>LAB:<T.MHRSLAB>"
"I NEED A SPECIAL ROOM  LECT, 1ST CHOICE:<T.RMLEC1>  2ND CHOICE:<T.RMLEC2>"
"                       LAB , 1ST CHOICE:<T.RMLAB1>  2ND CHOICE:<T.RMLAB2>"
"THERE ARE OTHER CONSTRAINTS, COMMENTS:  <T.COMMENT1>"
"<T.COMMENT2>"
"<T.COMMENT3>"
"NEXT COURSE <T.NXTCRS  SEGMENT <NXTSG> "
ON NOMATCH INCLUDE
ON NOMATCH IF NXTCRS EQ 'YYYYYY' GOTO TOP;
ON NOMATCH GOTO UPDCRS
ENDCASE
*-------------------------------------------------------------
CASE ADDPRF
TYPE "ENTRING CASEADDPRF"
COMPUTE
    SPNAME='XXXXXXXX';
    MAILC='WX';
    DEPT='00';
    YRQTR='813';
    COURSE='XX9999';
    SEGMENT='1';
    SPECSTUD='N';
    OTHPROF='N';
    TEAM='N';
    SPHOURS='000000000';
    SPDAY='000000';
    SPBLOLEC='1';
    SPBLOLAB='1';
    FINAL='Y';
    OTHCOURSE='XX9999';
    TOGCOURSE='XX9999';
    ACCEL='0';
    SPLIT='1';
    BACK='0';
    RMLEC1='X9990';
```

```
    RMLEC2='X9990';
    RMLAB1='X9990';
    RMLAB2='X9990';
    MHRSLEC='N';
    MHRSLAB='N';
    COMMENT1='0 COMMENT';
NXTCRS='XXXXXX';
PNAME=PNAME;
MATCH PNAME
ON MATCH REJECT
ON NOMATCH CRTFORM LINE 1
"FACULTY NAME: <T.PNAME> MAILCODE: <T.MC> DEPT NUMBER: <T.DP>"
"COURSE : <T.CN> YEAR AND QUARTER: <T.YQ>, SEGMENT<55<T.SG>"
"ACCELERATED/WEEK1-6  (1) OR REFRESHER/WEEK7-12 COURSE..?<T.ACCEL>"
"DO YOU NEED SPECIAL STUDENTS IN ONE SEGMENT.....? <55 <T.SPECSTUD>"
"ARE THERE OTHER FACULTY TEACHING SOME SEGMENT.....? <55 <T.OTHPROF>"
"I TEACH THIS COURSE IN TEAM WITH ANOTHER FACULTY....?<55 <T.TEAM>"
"*0...DON'T CARE  1...LIKE TO HAVE *  \SPECIAL HOURS?: <T.SPHOURS>"
"*2...DON'T LIKE  3...NEED TO HAVE *  /SPECIAL  DAYS?: <T.SPDAY>"
"THE FINAL IS SCHEDULED..:< T.FINAL> SPLIT LAB IN EQUAL SEGM.:<T.SPLIT>"
"DO NOT SCHEDULE AT SAME TIME WITH COURSE (NUMBER)...:<55 <T.OTHCOURSE>"
"DO SCHEDULE TOGETHER WITH COURSE (NUMBER)...:<55 <T.TOGCOURSE>"
"SCHEDULE BACK TO BACK (1), NOT BACK TO BACK (2)...:<55 <T.BACK>"
"I TEACH IN SPECIAL BLOCKS:  LECTURE:<47<T.SPBLOLEC>LAB:<55 <T.SPBLOLAB>"
"I NEED MORE/LESS HOURS THAN IN CATALOG: LEC:<47<T.MHRSLEC>LAB:<T.MHRSLAB>"
"I NEED A SPECIAL ROOM  LECT, 1ST CHOICE:<T.RMLEC1>  2ND CHOICE:<T.RMLEC2>"
"                       LAB , 1ST CHOICE:<T.RMLAB1>  2ND CHOICE:<T.RMLAB2>"
"THERE ARE OTHER CONSTRAINTS, COMMENTS:  <T.COMMENT1>"
"<T.COMMENT2>"
"<T.COMMENT3>"
"NEXT COURSE <T.NXTCRS  SEGMENT <NXTSG> "
ON NOMATCH INCLUDE
ON NOMATCH IF NXTCRS EQ 'XXXXXX' THEN GOTO TOP ELSE GOTO UPDCRS;
ENDCASE
*-------------------------------------------------------
DATA VIA FI3270
END
```

# APPENDIX G
# USER'S MANUAL FOR THE CONSTRAINT QUESTIONNAIRE

## The Constraint Questionnaire

This questionnaire program was written to support the computer data base about the scheduling process.

Among all the constraints faculty members express, the most common ones were collected and put together into a easy to handle program. This manual should help the user in filling out the questions.

Questions concerning this manual should be directed to the scheduler's office.

### *How to Start and Stop*

You will be supplied with programs running and supporting the questionnaire. They will reside on your disk as will the output of your work. To start, log on as usual. Then, type FOCUS (ENTER) and wait until you are in the program. You will know that you are there as soon as you see the words:
ENTER PROF NAME and
HIT ENTER TO CONTINUE.

There are several reasons to stop. Two kinds are discussed here:

1. Emergency Stop
   When you don't know what to do anymore and you want desperately to get out of the program do the following:
   Keep the ALT key down and hit the '1'(this combination is the PF1 key).
   As soon as there is no action on the screen anymore type FIN (ENTER).
   This should do the job.

If really nothing helps press down the ALT key and hit the PA1 key. You will see CP. Respond with I CMS (ENTER) and after you see some action on the screen type LOG (ENTER). This will do it for sure.

2. Regular stop
   You finished what you wanted to do and you want to stop now. Hit the ENTER key where ever you are in the program right now. The words 'ENTER PROF NAME' will appear on top of the screen.
   This time you enter END (ENTER) at this place and you are out of the program. Now type FIN (ENTER) and you are out of FOCUS.

*Report Results*

When you are ready with all your entries for this particular quarter and you want to report the results to the scheduler do the following: Type SF while in CMS. This brings up a Send File Exec. The file you want to send is CON FOCUS A, send it to the schedulers user ID. On request, you will be supplied with programs enableing you to get printouts of your input data.

*Description of the Individual Questions*

1. PROF NAME Write the last name and the initials of the first names
2. MAILCODE Write the two letters assigned to each faculty member.
3. DEPT NUMBER Write the department number of the faculty member
4. COURSE Enter the course number like 'AS1234' without blanks.
5. YEAR AND QUARTER Enter academic year and quarter like '853' meaning academic year 85, quarter 3.
6. SEGMENT You have to enter a segment number. If there is only one segment enter a '1'. If there is more than one segment you will have to answer a whole screen for each segment

although it might be taught by the same professor. Just keep numbering them in a successive fashion.

7. ACCELERATED If the course is a accelerated course or a refresher course then indicate it here.
   1...Course during first 6 weeks
   2...Course during second 6 weeks.
8. PARTIC. STUDENTS This question is related to the one before. Answer with a 'Y' or 'N'. If there is more than one segment and particular students should be together in one segment :'Y'. If this is not the case:'N'.
9. OTHER FACULTY If there are several segments and one or more other faculty members are teaching the same course but another segment in this quarter, answer 'Y'. Otherwise answer 'N'.
10. TEAM TEACH If the course is taught by more than one teacher at the same time (and the same segment) then answer 'Y'. Otherwise 'N'.
11. SPLIT LAB If for a course the lab part has to be held in several segments then input the number of lab segments. Default is 1; if the course is taught in several segments anyway, then an entry other than 1 will be treated as splitting every segment in equal parts.
12. TOGETHER COURSE Indicate the course number if there is a course to be scheduled together with this course (same time, same room).
13. BACK-TO-BACK If you would like to have your courses back-to-back (1) or not back-to-back (2) with the other courses you are teaching then indicate accordingly.
14. SPECIAL HOURS If this faculty member has any constraints about teaching at particular hours of the day, here is the place to write them down: The field consists of 9 entries, each representing one academic hour of the day. The first entry repre- sents the first hour, the second entry the second hour etc. If there is no constraint there should be a '0' in this place. Otherwise put the following:

1...means,he/she likes to teach at this hour

2...means,he/she does not like to teach at this hour

3...means,he/she needs to teach at this hour

Now fill the field with the appropriate number in the position of the hour wanted.

15. SPECIAL DAY The same codes apply here as for the previous one. Now the 6 entries represent the days of the week, starting with Monday and going through Saturday.

16. CONFLICT If there is a course the faculty member wants to take for credit, (or something similar) you should indicate it here by putting the number of this course into the field. Otherwise leave it as it is.(Don't put in the second or third course this prof is teaching; it is taken care of automatically)

17. SPECIAL BLOCK If the course is not taught 1 hour at a time, indicate here how many hours there should be in a row by putting in the number wanted. (if there is no lab or no lecture in this course, that's OK. Just leave the '1' in this field)

18. MORE/LESS HOURS It might be the case that there have to be more or less hours in this course than indicated in the current NPS catalog. Write down this number. (do this only if different from catalog.)

19. SPECIAL ROOM If a particular room is really needed, put in the room number and if possible a second choice. If NO room is needed, indicate by filling the whole field with zeros like '0000'. Otherwise leave the default: 'X999'.

20. OTHER CONSTRAINTS If there are constraints not covered by the questions before, or you have to say something in addition, do this in this three lines.

# APPENDIX H

## CON TABLE MASTER: FOCUS MASTER OF USED CONSTRAINT DATA

```
FILE=ALLREP          ,SUFFIX=FIX
SEGNAME=ALLREP
FIELDNAME    =COURSE        ,E01          ,A6         ,A08        ,$
FIELDNAME    =SEGMENT       ,E02          ,A1         ,A04        ,$
FIELDNAME    =TEAM          ,E03          ,A1         ,A04        ,$
FIELDNAME    =MAILC         ,E04          ,A2         ,A04        ,$
FIELDNAME    =SPECSTUD      ,E05          ,A1         ,A04        ,$
FIELDNAME    =OTHPROF       ,E06          ,A1         ,A04        ,$
FIELDNAME    =SPHOURS       ,E07          ,A9         ,A12        ,$
FIELDNAME    =SPDAY         ,E08          ,A6         ,A08        ,$
FIELDNAME    =FINAL         ,E09          ,A1         ,A04        ,$
FIELDNAME    =BACK          ,E10          ,A1         ,A04        ,$
FIELDNAME    =SPLIT         ,E11          ,A1         ,A04        ,$
FIELDNAME    =ACCEL         ,E12          ,A1         ,A04        ,$
FIELDNAME    =TOGCOURSE     ,E13          ,A6         ,A08        ,$
FIELDNAME    =OTHCOURSE     ,E14          ,A6         ,A08        ,$
FIELDNAME    =SPBLOLEC      ,E15          ,A1         ,A04        ,$
FIELDNAME    =SPBLOLAB      ,E16          ,A1         ,A04        ,$
FIELDNAME    =MHRSLEC       ,E17          ,A1         ,A04        ,$
FIELDNAME    =MHRSLAB       ,E18          ,A1         ,A04        ,$
FIELDNAME    =RMLEC1        ,E19          ,A5         ,A08        ,$
FIELDNAME    =RMLEC2        ,E20          ,A5         ,A08        ,$
FIELDNAME    =RMLAB1        ,E21          ,A5         ,A08        ,$
FIELDNAME    =RMLAB2        ,E22          ,A5         ,A08        ,$
FIELDNAME    =COMMENT1      ,E23          ,A30        ,A32        ,$
FIELDNAME    =COMMENT2      ,E24          ,A72        ,A72        ,$
FIELDNAME    =COMMENT3      ,E25          ,A72        ,A72        ,$
FIELDNAME    =PNAME         ,E26          ,A16        ,A16        ,$
```

## APPENDIX I

## EXERPT FROM THE CURRENTLY USED PRIORITY SETTING

**Listed in order of priority:**

1. Scheduling department and special meetings for professors, curriculum seminars for students, and Superintendent's lectures.
2. Refresher courses.
3. Special request for courses at required time.
4. Special requests for required times and days from professors.
5. Large courses.
6. Courses being taught by Chairmen and Deans.
7. Two courses that must be schedule at the same time.
8. Accelerated courses.
9. Courses with two or more segments requiring only designated students to be placed in one segment.
10. Courses requiring special rooms.
11. Courses requiring three lab hours.
12. Courses that use the same lab rooms and cannot conflict in time.
13. Courses with a large diversity of majors.
14. Professors with three or more classes or team teaching.
15. Professors traveling from long distances.
16. Regular courses.

# APPENDIX J

## EXAMPLE ILP, FORMULATION AND RESULT

```
MIN      XA1 + XA2 + XA3 + 200 XA4 + 400 XA6 + 200 XA7 + XC1
     + XC2 + XC3 + 200 XC4 + 400 XC6 + 200 XC7 + 200 XB1 + 2 XB2
     + XB3 + XB4 + XB6 + XB7 + 200 XD1 + 2 XD2 + XD3 + XD4 + XD6
     + XD7 + YA2 + 5 YA3 + YA4 + YA6 + 5 YA7 + YB1 + YB3 + YB4
     + YB6 + 25 YB7 + YC1 + YC3 + YC4 + YC6 + 60 YC7 + YD1
     + 45 YD4 + 90 YD6 + 180 YD7 + YE2 + YE3 + YE6 + YF1 + YF2
     + 20 YF3 + 10 YF4 + YF6 + YG3 + 15 YG4 + 30 YG6 + 60 YG7
     + YH2 + YH3 + YH6 + 30 YH7 + 110 YI1 + 55 YI2 + YI3 + YI6
     + 55 YI7 + YJ1 + YJ2 + 80 YJ3 + 40 YJ4 + YJ6 + YK2 + 15 YK3
     + YK4 + YK6 + 15 YK7 + YL2 + 10 YL3 + YL4 + YL6 + 10 YL7
     + YM1 + YM3 + 40 YM4 + YM6 + YN2 + YN4 + 70 YN6 + 140 YN7
     + YO2 + 10 YO3 + 10 YO4 + YO7
SUBJECT TO
     000)   XA2 + XA3 + XA4 + XA6 + XA7 =    1
     001)   XC2 + XC3 + XC4 + XC6 + XC7 =    1
     002)   XB1 + XB2 + XB3 + XB6 + XB7 =    1
     003)   XD1 + XD2 + XD3 + XD6 + XD7 =    1
     004)   YA2 + YA3 + YA4 + YA6 + YA7 =    1
     005)   YB1 + YB3 + YB4 + YB6 + YB7 =    1
     006)   YC1 + YC3 + YC4 + YC6 + YC7 =    1
     007)   YD1 + YD4 + YD6 + YD7 =    1
     008)   YE2 + YE3 + YE6 =    1
     009)   YF1 + YF2 + YF3 + YF4 + YF6 =    1
     010)   YG3 + YG4 + YG6 + YG7 =    1
     011)   YH2 + YH3 + YH6 + YH7 =    1
     012)   YI1 + YI2 + YI3 + YI6 + YI7 =    1
     013)   YJ1 + YJ2 + YJ3 + YJ4 + YJ6 =    1
     014)   YK2 + YK3 + YK4 + YK6 + YK7 =    1
     015)   YL2 + YL3 + YL4 + YL6 + YL7 =    1
     016)   YM1 + YM3 + YM4 + YM6 =    1
     017)   YN2 + YN4 + YN6 + YN7 =    1
     018)   YO2 + YO3 + YO4 + YO7 =    1
     019)   XA1 + XC1 <=   1
     020)   XA2 + XC2 <=   1
     021)   XA3 + XC3 <=   1
     022)   XA4 + XC4 <=   1
     023)   XA6 + XC6 <=   1
     024)   XA7 + XC7 <=   1
     025)   XB1 + XD1 <=   1
     026)   XB2 + XD2 <=   1
     027)   XB3 + XD3 <=   1
     028)   XB4 + XD4 <=   1
     029)   XB6 + XD6 <=   1
     030)   XB7 + XD7 <=   1
     031)   XA1 =    0
     032)   XB4 =    0
     033)   XC1 =    0
     034)   XD4 =    0
     035) - 17 XA1 - 17 XC1 - 17 XB1 - 17 XD1 + 5 YB1 + 12 YC1
     + 9 YD1 + YF1 + 11 YI1 + 4 YJ1 + 8 YM1 >=   0
     036) - 21 XA1 - 21 XC1 - 21 XB1 - 21 XD1 + 5 YB1 + 12 YC1
     + 9 YD1 + YF1 + 11 YI1 + 4 YJ1 + 8 YM1 <=   0
     037) - 17 XA2 - 17 XC2 - 17 XB2 - 17 XD2 + YA2 + YE2 + YF2
     + 6 YH2 - 11 YI2 + 4 YJ2 + 3 YK2 + 2 YL2 + 7 YN2 + 2 YO2  >=   0
     038) - 21 XA2 - 21 XC2 - 21 XB2 - 21 XD2 + YA2 + YE2 + YF2
     + 6 YH2 + 11 YI2 + 4 YJ2 + 3 YK2 + 2 YL2 + 7 YN2 + 2 YO2    <=   0
     039) - 17 XA3 - 17 XC3 - 17 XB3 - 17 XD3 + YA3 + 5 YB3
     + 12 YC3 + YE3 + YF3 + 3 YG3 + 6 YH3 + 11 YI3 + 4 YJ3
     + 3 YK3 + 2 YL3 + 8 YM3 + 2 YO3 >=   0
```

```
      040) - 21 XA3 - 21 XC3 - 21 XB3 - 21 XD3 + YA3 + 5 YB3
      + 12 YC3 + YE3 + YF3 + 3 YG3 + 6 YH3 + 11 YI3 + 4 YJ3
      + 3 YK3 + 2 YL3 + 8 YM3 + 2 YO3 <=    0
      041) - 17 XA4 - 17 XC4 - 17 XB4 - 17 XD4 + YA4 + 5 YB4
      + 12 YC4 + 9 YD4 + YF4 + 3 YG4 + 4 YJ4 + 3 YK4 + 2 YL4
      + 8 YM4 + 7 YN4 + 2 YO4 >=    0
      042) - 21 XA4 - 21 XC4 - 21 XB4 - 21 XD4 + YA4 + 5 YB4
      + 12 YC4 + 9 YD4 + YF4 + 3 YG4 + 4 YJ4 + 3 YK4 + 2 YL4
      + 8 YM4 + 7 YN4 + 2 YO4 <=    0
      043) - 17 XA6 - 17 XC6 - 17 XB6 - 17 XD6 + YA6 + 5 YB6
      + 12 YC6 + 9 YD6 + YE6 + YF6 + 3 YG6 + 6 YH6 + 11 YI6
      + 4 YJ6 + 3 YK6 + 2 YL6 + 8 YM6 + 7 YN6 >=    0
      044) - 21 XA6 - 21 XC6 - 21 XB6 - 21 XD6 + YA6 + 5 YB6
      + 12 YC6 + 9 YD6 + YE6 + YF6 + 3 YG6 + 6 YH6 + 11 YI6
      + 4 YJ6 + 3 YK6 + 2 YL6 + 8 YM6 + 7 YN6 <=    0
      045) - 17 XA7 - 17 XC7 - 17 XB7 - 17 XD7 + YA7 + 5 YB7
      + 12 YC7 + 9 YD7 + 3 YG7 + 6 YH7 + 11 YI7 + 3 YK7 + 2 YL7
      + 7 YN7 + 2 YO7 >=    0
      046) - 21 XA7 - 21 XC7 - 21 XB7 - 21 XD7 + YA7 + 5 YB7
      + 12 YC7 + 9 YD7 + 3 YG7 + 6 YH7 + 11 YI7 + 3 YK7 + 2 YL7
      + 7 YN7 + 2 YO7 <=    0
END
INTEGER-VARIABLES=  91


    LP OPTIMUM FOUND  AT STEP    56


    ENUMERATION COMPLETE. BRANCHES=    17  PIVOTS=   462

LAST INTEGER SOLUTION IS THE BEST FOUND
 RE-INSTALLING BEST SOLUTION...

         OBJECTIVE FUNCTION VALUE

            108.000000


VARIABLE          VALUE          REDUCED COST
      XA2          1.000000          1.000000
      XC3          1.000000          1.000000
      XB6          1.000000          1.000000
      XD3          1.000000          1.000000
      YA6          1.000000          1.000000
      YB3          1.000000          1.000000
      YC3          1.000000          1.000000
      YD6          1.000000         90.000000
      YE2          1.000000          1.000000
      YF2          1.000000          1.000000
      YG3          1.000000          1.000000
      YH3          1.000000          1.000000
      YI6          1.000000          1.000000
      YJ2          1.000000          1.000000
      YK2          1.000000          1.000000
      YL2          1.000000          1.000000
      YM3          1.000000          1.000000
      YN2          1.000000          1.000000
      YO2          1.000000          1.000000
```

## LIST OF REFERENCES


1. Naval Postgraduate School Technical Note No.0211-01 *The Heuristic Academic Master Scheduler*, by J. Bow, 1966.

2. Tripathy, A., "School Timetabling - A case in Large Binary Integer Linear Programming," *Management Science*, V.30, No.12, pp.1473-1489, 1984.

3. Dempster, M.A.H., Lethbridge, D.G., and Ulph, A.M., "School Timetabling by Computer - A Technical History," *Educational Research*, V.18, No.1, pp.24-31, 1975.

4. Dowsland, W.B., "Computer Aided School Timetabling, Part1: The History of Computerized Timetabling," *Computer Education*, No.42, pp.22-23, November 1982.

5. Tripathy, A., "A Lagrangean Relaxation Approach to Course Timetabling," *Journal of the Operational Research Society*, V.31, pp.599-603, 1980.

6. Selim, S.M., "An Algorithm for Constructing a University Faculty Timetable," *Computer and Education*, V.6, No.4, pp.323-332, 1982.

7. Selim, S.M., "An Algorithm for Producing Course and Lecturer Timetables," *Computer and Education*, V.6, No.2, pp.101-108, 1983.

8. Almond, M., "An Algorithm for Constructing University Timetables," *Computer Journal*, V.8, pp.331-340, 1966.

9. Romero, B.P., "Examination Scheduling in a large Engeneering School: A Computer-assisited Participative Procedure," *Interfaces*, V.12, No.2, pp.17-23, April 1982.

10. Mehta, N.K., "Computer Based Examination Management System," *Journal of Educational Technology Systems*, V.11, No.2, pp.185-198, 1982-83.

11. *Catalog Academic Year 1984*, Naval Postgraduate School, 1984.

12. Schrage, L., *Linear, Integer, and Quadratic Progamming with LINDO*, The Scientific Press, 1984.

## BIBLIOGRAPHY


Aho, A.V., Hopcroft, J.E., and Ullman, J.D., *The Design of Computer Algorithms*, Addison-Wesley Publishing Company, 1974.

Braham, A.M., Westwood, J.B., "A simple Heuristic to Facilitate Course Timetabling", *Journal of Operational Research of England*, V.29, pp. 1055-1060, 1980.

Barraclough, E.D., "The Application of a Digital Computer to the Construction of Timetables", *Computer Journal*, V.8, pp.136-146, 1965.

Croucher J.S., "A Goal Programming Model for Timetabling," *Journal of Operational Research of India*, V.21, No.3, pp.145-152, 1984.

Davis, T.I., "School Timetabling by Computer," *Trends in Education*, V.33, Oct 1983.

Ferland, J.A., Roy, S., "Timetabling Problem for University as Assignment of Activities to Resources", *Computer and Operations Research*, V.12, No.2, pp.207-218, 1985.

Larie, N.L., "Integer Linear Programming for a Timetabling Problem", *Computer Journal*, V.12, pp.305-315, 1969.

Lions, J., "The Ontario School Scheduling Program", *Computer Journal*, V.10, pp.14-21, 1966.

Papoulias, D.B., "The Assignment-to-day problem in a School Time-table, a Heuristic Approach", *European Journal of Operational Research*, V.4, pp.31-41, 1980.

Report of the Curriculum and Timetable Strategies Preliminary Research Project, *The Use of Computer Assistance in Curriculum Planning*, by Pratt, S., August 1978.

University of Montreal, Dept. of Information and Operations Research, Publication No.531, *The Timetabling Problem*, by Ferland, J.A., Roy, S., Loc, T.G., May 1985.

Waterloo University (Ontario), Dept. of Management Science Report WP-14, *Construction of School Timetables by Flow Methods*, by de Werra, D.,1970.

## INITIAL DISTRIBUTION LIST

| | | No. Copies |
|---|---|---|
| 1. | Defense Technical Information Center<br>Cameron Station<br>Alexandria, Virginia 22304-6145 | 2 |
| 2. | Library, Code 0142<br>Naval Postgraduate School<br>Monterey, California 93943-5100 | 2 |
| 3. | Dean of Academic Administration<br>Code 014<br>Naval Postgraduate School<br>Monterey, California 93943-5100 | 1 |
| 4. | Department Chairman, Code 55<br>Department of Operations Research<br>Naval Postgraduate School<br>Monterey, California 93943-5100 | 1 |
| 5. | Professor R. Kevin Wood<br>Code 55wd<br>Department of Operations Research<br>Naval Postgraduate School<br>Monterey, California 93943-5100 | 10 |
| 6. | Professor R. Rosenthal<br>Code 55ro<br>Department of Operations Research<br>Naval Postgraduate School<br>Monterey, California 93943-5100 | 1 |
| 7. | Professor James N. II Eagel<br>Code 55er<br>Department of Operations Research<br>Naval Postgraduate School<br>Monterey, California 93943-5100 | 1 |
| 8. | Scheduler, Code 0144<br>Naval Postgraduate School<br>Monterey, California 93943-5100 | 1 |
| 9. | Informations Systems, Code 146A<br>Naval Postgraduate School<br>Monterey, California 93943-5100 | 1 |
| 10. | Captain Dietmar W. Fiegas<br>1./ Instandsetzungsbataillon 610<br>Briesen Kaserne<br>2390 Flensburg<br>Federal Republic of Germany | 4 |

END

FILMED

1-86

DTIC