Student scheduling: A solution method for the conflict matrix

Beverly Leo Higginbotham

The University of Montana

Let us know how access to this document benefits you.
Follow this and additional works at: https://scholarworks.umt.edu/etd

Recommended Citation
https://scholarworks.umt.edu/etd/8331

This Thesis is brought to you for free and open access by the Graduate School at ScholarWorks at University of Montana. It has been accepted for inclusion in Graduate Student Theses, Dissertations, & Professional Papers by an authorized administrator of ScholarWorks at University of Montana. For more information, please contact scholarworks@mso.umt.edu.
STUDENT SCHEDULING:

A SOLUTION METHOD FOR THE CONFLICT MATRIX

By

Beverly L. Higginbotham

B.S., Carnegie-Mellon University, 1966

Presented in partial fulfillment of the requirements for the degree of

Master of Business Administration

UNIVERSITY OF MONTANA

1970

Approved by:

Chairman, Board of Examiners

Dean, Graduate School

Date
ACKNOWLEDGMENTS

I am indebted to Dr. Bernard J. Bowlen and Mr. Carl J. Schwendiman for their constructive criticism during the preparation of this professional paper. Mr. Edward A. Peressini of The College Of Great Falls contributed much to the mathematical content of this paper.

I am grateful to Grace Molen for her unfailing and efficient assistance in typing and proofing this work.

My greatest debt is to Susan, my wife. Without her patience and encouragement, this work would not be possible.

Finally, I would like to thank the Strategic Air Command and the Air Force Institute Of Technology of the United States Air Force for providing the opportunity to complete this paper.
# TABLE OF CONTENTS

**Chapter**

I. INTRODUCTION ........................................... 1  
   The General Problem Setting  
   Problems Encountered in Constructing the  
   Master Schedule  
   Research Problem  
   Research Objective  
   Procedures to be Used

II. REVIEW OF THE LITERATURE ............................ 7  
   Manual Scheduling  
   The Unit Record Approach  
   Computer Scheduling  
   Data Collection and Student Assignment Programs  
   Generating and Evaluating the Master Schedule  
   Heuristic Schedulers  
   A Mathematical Scheduler

III. RAPID APPROXIMATION METHOD - A PROPOSED SOLUTION METHOD .................................. 22  
   Scheduling the University of Montana MBA Program at Malmstrom Air Force Base  
   The Conflict Matrix  
   Use of the Conflict Matrix  
   The Rapid Approximation Method (RAM)  
   Expanding the Usefulness of RAM  
   Computerized RAM

IV. TESTING THE METHOD ................................. 51  
   Two Practical Tests  
   Four Hypothetical Tests

V. SUMMARY ............................................. 63  
   Results of Testing RAM  
   Limitations of the RAM Method  
   Conclusions  
   Recommendations for Further Research

APPENDIX ................................................ 69

SOURCES CONSULTED .................................... 74

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
# LIST OF ILLUSTRATIONS

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. The Conflict Matrix</td>
<td>24</td>
</tr>
<tr>
<td>2. The Complete Conflict Matrix</td>
<td>29</td>
</tr>
<tr>
<td>3. An Illustration of RAM</td>
<td>33</td>
</tr>
<tr>
<td>4a. Conflict Between Three Courses. No student requesting all three</td>
<td>38</td>
</tr>
<tr>
<td>4b. Conflict Between Three Courses. One student requesting all three</td>
<td>38</td>
</tr>
<tr>
<td>5. Three Dimensional Conflict Matrix</td>
<td>40</td>
</tr>
<tr>
<td>6. RAM Flowchart</td>
<td>42</td>
</tr>
<tr>
<td>7. RAM Computer Program</td>
<td>44</td>
</tr>
<tr>
<td>8. RAM Input Data</td>
<td>48</td>
</tr>
<tr>
<td>9. Evaluation of RAM</td>
<td>57</td>
</tr>
<tr>
<td>10. Evaluation of RAM</td>
<td>60</td>
</tr>
<tr>
<td>11. Evaluation of RAM</td>
<td>61</td>
</tr>
<tr>
<td>12. Evaluation of RAM</td>
<td>62</td>
</tr>
</tbody>
</table>

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
CHAPTER I

INTRODUCTION

The General Problem Setting

The Administrator's Dilemma

In recent years school administrators at all levels have been faced with sky-rocketing enrollments, rapidly expanding and changing curriculum requirements, a shortage of qualified faculty, and a scarcity of available classroom space. Nowhere are the results of the changes felt more dramatically than in our secondary and collegiate level schools. Administrators now find greater demands than ever before leveled upon their time and talents. They must make long range plans for upgrading facilities. They are attempting to implement new teaching methods and desirable curriculum changes, and like it or not, they must cope with an ever expanding student body in physical plants which are rapidly becoming over-crowded and under-staffed.

As a result of these pressures, school officials are faced with two conflicting goals. On one hand, administrators seek expanded and revised curriculums for a growing student population. And on the other, they are hard
pressed to implement their current curriculums given the limiting constraints of their present facilities. The time demands of scheduling current classes seriously detract from the time available for revising and improving present curriculums. The purpose of this paper is to reduce the time required to construct a master schedule by providing a useful decision method, and in so doing, permit the administrator to tackle the more pressing problem of curriculum development.

Problems Encountered in Constructing
The Master Schedule

All aspects of education, no matter how far removed from the actual learning situation, have a bearing upon the whole of education. One of the most difficult and time consuming problems for the school administrator is efficiently allocating the resources at his disposal in constructing the master schedule. At best, it is a tedious, frustrating, and time consuming process. "For a medium sized high school it takes upwards of 1,000 man-hours, including a high percentage of expensive and scarce administrative time."¹ Much of the work is sheer clerical drudgery: the listing of subjects, rooms and instructors;

the tallying of student course requests; and the making of a conflict matrix. But, once these menial tasks have been completed, it remains for the administrator to fit the pieces of the puzzle together to form the best possible schedule. In so doing, he must be careful to avoid conflicts or resolve them for the greatest good for students and teachers.

The Research Problem

Minimizing Student Conflicts in the Master Schedule

Each year numerous school administrators spend untold hours wrestling with the thorny task of constructing the master schedule for their schools. They are constrained by limited facilities and intimidated by ever expanding enrollments. In order to efficiently allocate the resources at their disposal, they must produce a schedule of classes which contains as few conflicts as possible.

A number of different conflicts arise in attempting to construct the master schedule. The first is assigning one instructor to teach two courses during the same period, because no other instructor is available. The second involves scheduling two courses into the same room for one period. Obviously these two types of conflicts can not be permitted in the final master schedule.

The third type of conflict involves student conflicts. A conflict arises when a student desires to
take two single-section courses which are offered during the same period. For example, assume that Biology and Latin 2 are single-section courses. If a student requested both courses, and if they were offered during the same time period, this would count as one conflict.

The conflict matrix is a compilation of all conflicts for all students. It lists the number of students desiring to take any combination of two single-section courses.

These single-section courses are usually the most difficult to fit into the master schedule. When two such courses are scheduled during the same period, certain students will be denied the opportunity to enroll in both. Therefore it is imperative that the master schedule be designed to reduce such conflicts to a minimum.

In a small school which has only eight single-section courses to be scheduled during a four period day there are one hundred and five different possible schedules. Any one of these schedules could be the optimal solution. To find the schedule having the fewest conflicts, the administrator could list all possible solutions and then choose the best. But in a school where there are sixteen single-section courses to be scheduled, the number of possible combinations is over two million. Obviously the administrator needs a method which will give him the solution more efficiently than the enumeration method.
There is no method currently available which indicates which single-section courses should be scheduled concurrently in order to minimize student conflicts. In order to meet student requests, many schools schedule course sections throughout the day. This attempt to reduce single-section conflicts results in some sections being less than full and is therefore wasteful in terms of teacher and classroom utilization.

The problem to be attacked is minimization of student conflicts for single-section courses in the final master schedule.

**Research Objective**

Efficient Use of School Resources

The objective of this research is to find a method to schedule classes in order to minimize the number of student conflicts. In this manner, more efficient use can be made of school resources. This research will deal with only a small portion of the problem of building an efficient master schedule. It will embrace the area of student conflict avoidance for single-section courses in the final master schedule.

**Procedures to be Used**

**Organization of the Paper**

A review of the literature pertaining to student scheduling is contained in Chapter II. The problems
encountered in scheduling single-section courses and a proposed solution are presented in Chapter III. In Chapter IV the proposed solution method is tested as a practical technique. The results of these tests and limitations on the method are presented in the final chapter. Additionally, the conclusions reached and recommendations for further research are outlined in Chapter V.
CHAPTER II

REVIEW OF THE LITERATURE

The problem of resolving single-section conflicts in school scheduling is part of the larger task of designing an acceptable master schedule. As such, any discussion of conflict avoidance must necessarily include a review of various approaches to the problem of generating a master schedule. A discussion of many approaches will help to crystallize the various problems encountered. The survey will start with the earliest manual techniques and proceed chronologically to the most recent computerized mathematical methods.

Manual Scheduling

Prior to the use of data processing equipment and later computers, the scheduling process was done entirely by hand. At best, this was a tedious and time consuming task. Before the administrator can design a master schedule, he needs information concerning student course requests, classroom space, and teacher availability.

The first step in collecting information is the tallying of student requests in order to determine the number of sections required for each course. Next, a
conflict matrix is developed. This matrix reflects the number of students desiring to take any combination of two courses. Additionally, information on the number of classrooms and the seating capacity of each is required. Here, consideration must be given to the use of specialized classrooms which can be used only for certain classes, for example, homemaking and woodworking rooms. Finally, the scheduler requires information concerning the courses which individual instructors are qualified to teach.

The vast number of variables involved demands a system to insure that all items are considered. One of the earliest such accounting systems involved using a scheduling board. This board was divided into rectangles with each area representing a period in the school day. Within each period, hooks representing available classrooms were attached to the board. Additionally, tags representing each teacher and each class section were prepared.

Once the data and a method for handling it were developed, the administrator was prepared to combine the resources at his disposal into an initial schedule. David B. Austin and Noble Gividen¹ suggest the following approach:

¹David B. Austin and Noble Gividen, The High School Principal and Staff Develop the Master Schedule, Bureau of Publication, Teachers College, Columbia U., New York, 1960, p. 58.
"1. Locate the single and double section course tags in periods which will allow for the fewest conflicts among them.

2. Now locate all other class tags as to periods.

3. Adjust the section tags. Hold them in the same designated periods, but shift them among teachers and rooms so that the best teacher programs will be completed, and reasonable use of space and equipment provided.

4. Check for internal consistency. In addition to a reasonable assignment for each teacher, it is well to examine the first tentative schedule, period by period, to estimate the adequacy of the program relative to various student groupings.

5. Test with sample programs for students. Seek conflicts and adjust the schedule to accommodate them."

With this tentative schedule in hand, the administrator will begin the task of assigning students to the courses which they had requested. Without fail this will involve great numbers of conflicts, over loaded or imbalanced sections, and in general, an unacceptable schedule. This, in turn, would necessitate revisions to the master schedule, assignment changes, reevaluation of the new schedule, further schedule adjustments, and so it would go until a workable master schedule was finally produced.

These first attempts at preparing a master schedule were time consuming and administrators quite obviously did not have time to prepare a great number of alternatives in an attempt to find a "better" schedule.
However, this initial experience with school scheduling did identify the three basic phases which are inherent in all student scheduling operations:

- **Phase 1**: Collection and sorting of input data.
- **Phase 2**: Generation and evaluation of a master schedule.
- **Phase 3**: Assignment of students to classes.

As the art of student scheduling evolved, the old schedule boards gave way to data processing cards and the tags were replaced by words written in computer memories. At first, the improvements tended to be concentrated in one or another of the three phases. Then, as researchers became aware of work done in other phases, whole systems were developed to handle the entire process from start to finish.

**The Unit Record Approach**

The advent of mass data processing equipment brought a reduction in the amount of time needed to produce a good class schedule. The unit record approach was nothing more than an extension of the manual method of compiling and sorting data. Tallying of student requests and the building of conflict matrices was achieved through collating machine listings of punch card information. The principal and his staff still had to construct the master
schedule, but once this was completed, successive sorting and collating during the assignment phase enabled them to evaluate their schedules much more rapidly.

The major drawbacks of the unit record approach were the amount of time required for the manual construction of the master schedule and the necessity for a separate read-sort run each time a different set of course combinations was attempted in the master schedule.²

Computer Scheduling

Following the unit record era, the computer attained a measure of acceptance in some of the larger school systems. As school officials became familiar with the capabilities of their new administrative partner, they began to search for applications unique to their environment in which the computer could assist them. The frustrating task of student scheduling was a prime candidate.

During the mid 1950's, computer memory and logic limitations precluded its use in the complicated task of constructing a master schedule. The machine's capabilities obviously pointed toward applying its high speed processing and printing ability to the data collection and assignment phases of student scheduling.

²Student Scheduling System / 360, Application Description, IBM Corp., 1966, p. 11.
Data Collection and Student Assignment Programs

One of the first workable programs of the data-collection-assignment variety was developed at Purdue University in 1956 by James Blakeley. Shortly thereafter, IBM developed a scheduling program called CLASS based upon the exploratory work done at Purdue. The program assumes a fixed schedule of classes, times, rooms, and teachers assigned by the traditional manual method. Combining this input information with student course requests, CLASS produces (1) a tally of course requests, (2) a conflict matrix, and (3) individual student schedule printouts.

Since that time, several refinements have been made to the earlier assignment programs which increase their usefulness as an aid to the manual construction of the master schedule. These newer software packages perform additional tasks such as printing class lists, and class cards, generating grade reports, and so forth. Today, there are a number of assignment programs available. IBM alone has several. Three of these are their STUDENT program for the 1620 series computer, SOCRATES for the 1400 series,

---


and Student Scheduling System/360 for the 360 series machines. Additionally, Control Data, National Cash Register and Honeywell all have assignment programs available for their hardware systems.

The assignment programs are generally suited for large schools which have up to twenty sections of a single course and as many as 20,000 students. Their ability to rearrange scheduling information into a useful format have made them invaluable to the administrator. Even with the aid of assignment programs, however, the administrator must still construct the master schedule by hand. Although the unit record and assignment programs perform clerical operations well, and tally information rapidly, the scheduler must still rely on his own genius for the actual construction of a master schedule. Somehow he has to efficiently allocate the resources at his disposal and produce a workable schedule.

Generating and Evaluating the Master Schedule

In 1964, N. L. Boyles developed a numeric code system which identified teachers, rooms, and time periods to the computer. Using these codes he devised the following set of heuristic decision rules to aid in the construction of the master schedule:

---

"1. Schedule classes so that the subject code and the room code are equal.

2. Place single-section subjects in the schedule matrix after fixed time activities and before multiple section subjects, and place all conflicting single-section subjects in the matrix at a time period other than one opposing the conflicting single-section subject.

3. Schedule a single-section subject opposite a subject being offered in multiple sections, or opposite a subject which has not been chosen by any of the students choosing the original single-section subject.

4. Do not schedule a two-section subject opposite two single-section subjects.

5. Do not schedule a three-section subject opposite two single-section subjects.

6. Schedule a subject with four or more sections opposite any subject regardless of the number of sections in the opposing subject. However, do not schedule multiple sections of the same subject at the same time."

Heuristic Schedulers

In the early 1960's two computerized methods which simulate manual scheduling techniques were introduced. These approaches do not completely take over the job of the scheduler. Rather, they are tools which speed the production of schedules.

The first of these simulation tools is GASP. It was developed at Massachusetts Institute of Technology, and follows a set of preprogrammed decision rules in order to arrive at a workable master schedule.

6Judith Murphy, op. cit., p. 8.
First the computer is fed information on the number of sections in each course, the number of students allowed in various sections, the number of times per week each section is to meet, faculty available, rooms available, and room capacity. In addition, certain courses may be specified for a particular room or time.

The program then attempts to assign the course sections to an appropriate combination of time, space and instructor, so that no conflicts arise among these variables. If a conflict is encountered, the program will attempt another random assignment. It will continue to loop until a no conflict schedule is found. If a particular course cannot be scheduled in a specified number of loops, the program goes on to the next course. Any unscheduled courses are later assigned by changing their faculty or room specification.

The program then reads a student's course requests and attempts to assign him to a section. If a particular section is full, it attempts to find an alternate section of the same course. If all sections of a course are closed or if a student cannot be scheduled due to a time conflict, that course is assessed "penalty points," and the program goes on to the next student.

After an attempt is made to schedule all students, penalty points for all courses are tallied. This figure and the initial master schedule are stored on tape. The master
schedule in core is then adjusted manually in an effort to alleviate some of the conflicts and another assignment run is made. If the adjusted schedule produces fewer penalty points, it replaces the schedule on tape. The assignment adjustment cycle is reiterated until an acceptable schedule is devised.

Stanford University also developed a heuristic schedule under the direction of Professor Robert Oakford. It does not follow precisely the GASP heuristic method; however, many elements are common to both. The major difference involves the number of elements of the scheduling problem each will accept. GASP can schedule as many as 4,000 courses and an unlimited number of students compared with Stanford's limit of 500 courses and 3,000 students. Stanford's advantage lies in its use of the modular scheduling concept. By breaking the school day into twenty minute periods, rather than the traditional sixty minute length, the program is able to incorporate such innovations as short group discussions, long laboratory periods, and independent student study in the schedule. Additionally, Stanford's program also provides for team teaching and specifying two alternatives for each course for each student.

---


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Because of the advantages of the Stanford program, it seems to have replaced the GASP scheduling methods in many schools.

**A Mathematical Scheduler**

In the past few years, educators and computer personnel have become interested in applying linear programming to the task of building a master schedule. Robert E. Harding developed the first feasible theoretical model and later F. T. Helmers incorporated the model in a practical application.

Harding viewed the problem of constructing a master schedule as a two-dimensional matrix. Each of the columns represented a time period during the day. Each row denotes a course. The intersection of a row and a column represented teaching the ith course in the jth time period if the element has the value 1. If the course is not scheduled for the jth time period, the value is zero.

The scheduler is faced with many restrictions when he attempts to construct the schedule. The linear programming model simulates some of these restrictions by defining

---


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
constraint equations. If $x_{ij}$ is defined as the value of the
ith course in the jth time period (either 0 or 1), then the
following constraints can be constructed.

1. A given course can meet no more than once in
   a given time period: $x_{ij} \leq 1$

2. A specific course section must meet T times
   per schedule cycle: $\sum_{j=1}^{J} x_{ij} = T$

3. A particular course is not to meet in a given
time period: $x_{ij} = 0$

4. Certain courses must meet in a particular
   period: $x_{ij} = 1$

5. Because of student conflicts, single-section
courses (say $x_{2j}$ and $x_{15j}$) must not meet during the
   same period: $x_{2j} + x_{15j} \leq 1, j = 1, 2, \ldots, J$

After these and other constraint equations have
been incorporated in the model, Harding offers two basic
solution methods. One approach is to simply solve the
generated system to find the course-section, time period
relations (the $x_{ij}$'s) which do not violate any of the given
constraints. Or the objective function,

$$\sum_{i=1}^{I} \sum_{j=1}^{J} c_{ij} x_{ij}$$

where $c_{ij}$ is some "figure of merit" for scheduling the ith
course to the jth period, can be minimized.
The model was field tested by Helmers\textsuperscript{10} at the Turtle Creek High School in Pennsylvania and the approach yielded acceptable results. However, all $c_{ij}$'s were assigned a value of 1. A school having one hundred courses to be scheduled during an eight period day would require a basic tableau containing eight hundred elements. The administrator is required to assign a subjective "figure of merit" to each of these elements. Because of the large number of $x_{ij}$'s, it would have been next to impossible to assign meaningful values to corresponding $c_{ij}$'s.

The linear programming approach has several advantages over the heuristic schedulers. First, it will find any and all solutions and if none exist it will quickly relate this. In contrast, the heuristic method may not relate that a solution does not exist under the given restrictions. Second, linear programming will provide shadow price information. With the aid of these prices, the administrator can evaluate the incremental returns associated with contemplated changes in student course demands, physical plant size, and instructor resources.

In spite of these obvious advantages, the linear programming approach has some significant drawbacks. The schedule generated must be the same each day of the week. There is also a need to easily identify and express

\begin{figure}
\centering
\includegraphics[width=\textwidth]{figure}
\caption{Diagram of schedule}
\end{figure}

\textsuperscript{10}Ibid, p. 42.
mathematically, the interrelationships between school policies and the resultant constraint structure. Last, and most important, the sheer size of the model has deterred schools which do not have access to a computer, from implementing the method.

In 1968, Robert Voght devised a new method for expressing the conflict matrix as part of a modular scheduling system for the Florida State University School.\(^\text{11}\) The concept involves expressing conflicts in terms of probability. It utilizes an index number, expressed as a per cent, to indicate the possibility of a student in a given course having a conflict with another course. Voght then uses these index numbers to aid in the construction of the master schedule.

Each of the scheduling systems discussed in this chapter deals with the problem of single-section courses and student conflicts in one of two ways. The heuristic schedulers such as GASP, try various random combinations of courses in an attempt to find some schedule with an acceptable number of conflicts. The alternate method of dealing with single-sections is simply to disallow any schedule which has two conflicting single-section courses in the same period. This approach is incorporated in the

methods proposed by Austin and Gividen, Boyles, and Harding. When attempting to use these methods the question arises; how should single-section courses be scheduled when the number of conflicting single-section courses is greater than the number of periods in the school day, i.e., when some single-section course must be scheduled opposite other single-section courses?

In an attempt to answer this question the author has developed an algorithm which schedules single-section courses so that the total number of conflicts in the schedule tends to be minimized. This algorithm is presented in the following chapter.
CHAPTER III

RAPID APPROXIMATION METHOD
A PROPOSED SOLUTION METHOD

Introduction

An algorithm is presented which schedules single-section courses so that the total number of student conflicts tends to be minimized. First, an illustration of the type of scheduling problem for which the algorithm was developed is presented. Next, the traditional methods of resolving conflicts are presented together with a discussion of some problems encountered in their application. Then, the algorithm developed by the author is introduced, coupled with an exploration of the principles upon which it is based. Finally, a number of modifications to the algorithm which make it useful as a practical tool are discussed. The last section of this chapter contains a computer program which may be used to apply the proposed algorithm.

Scheduling the University of Montana
MBA Program at Malmstrom Air Force Base

The method for scheduling students presented in this paper is the direct result of the unique scheduling
requirements for the University of Montana MBA program at Malmstrom Air Force Base. The first requirement is that all courses must be single-sections. Second, classes must meet once every fifth day. Third, since eight courses are offered each quarter and there are only five faculty members available, some professors must teach two courses. Fourth, students may enter the program at the beginning of any quarter and be given transfer credit for all undergraduate courses and up to twelve graduate course credits. This yields a student body with diverse combinations of course requests for any given quarter. A constraint found in usual student scheduling problems is absent at Malmstrom. Classrooms are adequate for current enrollment.

Considering the constraints of single-section courses, combined with diverse student course requests, the scheduling problem becomes one of minimizing rather than eliminating student conflicts. Since many other academic institutions are faced with this same problem, the author feels that the method presented in this paper may be useful to others.

The Conflict Matrix

It was noted in Chapter II that all scheduling systems took student conflicts into account in one way or another. One of the simplest ways to present the number of student conflicts is the conflict matrix.
Figure 1 illustrates a typical conflict matrix. Each row and each column represent one course. The intersection of a row and column shows the number of conflicts between the two courses. For instance, if the scheduler were considering scheduling course numbers three and five during the same period, he could readily determine that four students would have a conflict.

<table>
<thead>
<tr>
<th>Course Number</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td>15</td>
<td>5</td>
<td>11</td>
<td>8</td>
<td>1</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>X</td>
<td>19</td>
<td>6</td>
<td>10</td>
<td>11</td>
<td>11</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td>X</td>
<td>6</td>
<td>4</td>
<td>12</td>
<td>9</td>
<td>15</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>12</td>
<td>1</td>
<td>4</td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>9</td>
<td>4</td>
<td>9</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>12</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>12</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

Fig. 1.—The Conflict Matrix
Use of the Conflict Matrix

In order to illustrate how the conflict matrix can aid in the construction of a master schedule, let us assume a four period day. There are eight courses, therefore two courses will be taught during each period. The scheduler generally tries various combinations of courses until he arrives at some satisfactory schedule. In the example matrix, the scheduler could determine with some effort that teaching courses two and eight during one period, three and five during a second period, four and seven in a third, and one and six during the remaining period will result in a schedule which minimizes the total number of conflicts.

In making this determination, the scheduler could use one of two approaches. The first approach requires that every distinct combination or schedule be evaluated, and that the one which minimizes total conflicts be chosen. In this simple example there are one hundred and five possible schedules. But the scheduling of eight courses is a small problem by most standards. As the number of courses increases, the total number of distinct combinations becomes astronomical.
The number of combinations can be expressed by the following equation:

\[ C_d = \frac{n!}{(n/p)! (p!)} \]

where \( C_d \) = number of distinct combinations
\( n \) = the number of courses to be scheduled
\( p \) = the number of periods in the school day.

(See Appendix A for derivation.)

To illustrate, suppose we have a school with eighteen single-section courses to be taught in a nine period day. Using the above equation, we find that there are

\[ C_d = \frac{18!}{(18/9)! (9)!} = 3.44 \times 10^7 \]

or 34,400,000 distinct scheduling combinations to be evaluated. A task of this size would take an IBM 1620 computer approximately twenty-seven days. Obviously, complete enumeration of scheduling combinations is not the answer.

The other alternative available to the scheduler is the sampling approach. This technique may involve nothing more than solution by inspection, i.e., examining the conflict matrix and choosing some schedule which tends to have minimum conflicts. This approach is satisfactory when a small number of courses are involved, or when only two courses are to be taught per period. When the scheduling
problem grows beyond these bounds, however, solution by inspection becomes an unmanageable task.

The following sequential sampling approach eliminates this problem (and also the problem of complete enumeration). It involves selecting a random sample of schedule combinations and choosing the one with the smallest conflict. If the selected combination indicates an acceptable level of total conflict, the scheduler uses that schedule. If the conflict level is unacceptable, the scheduler determines the amount by which the total conflicts are expected to decrease if another set of random schedules were evaluated.\(^1\) If the value of a schedule containing fewer conflicts is greater than the cost of increasing the sample size, another set of schedules is evaluated. This process is reiterated until either an acceptable conflict level is achieved or until the cost of evaluating one more set of schedules is greater than the amount by which total conflicts are expected to decrease.\(^2\)

This approach has the advantage of indicating when it is no longer "profitable" to increase the sample

---

\(^1\)Given information about the distribution of conflicts in schedules previously evaluated, the administrator can make a subjective judgement concerning the probabilities of finding a schedule containing fewer conflicts in future samples.

size. Difficulty is encountered, however, in employing the technique because there is no exact method of expressing the worth of decreasing the number of conflicts.

Because of the difficulties encountered in the above approaches to scheduling single-section courses, the author has developed an algorithm called RAM (Rapid Approximation Method). RAM seeks a schedule of courses which will tend to minimize the total number of conflicts for single-section courses. The algorithm has the following features.

1. Given a list of single-section courses, teaching assignments, and student course requests, RAM finds a solution set which tends to minimize total conflicts.

2. The method requires a finite number of iterations equal to the number of periods in the school day.

3. The solution set does not indicate in which period the combination should be taught. This is left to the scheduler.

4. The resulting schedule is not necessarily optimal in all cases. Several heuristic decision rules are built into the method, however, which give it a high probability of finding the schedule which is optimal or nearly optimal.
The Rapid Approximation Method

The conflict matrix is illustrated in Figure 1 in its usual form. Because the matrix is symmetrical, the lower left portion is generally omitted. However, for the purpose of this discussion the lower left is included as shown in Figure 1. In this form each row or each column shows the conflict of one course with all other courses. For instance, suppose we are interested in finding the conflicts for all courses with course number four. Using Figure 1, the conflicts of courses one, two, and three with four could be found in column four. The remaining conflicts with courses five through eight can be found in row four. By using Figure 2, the same can be accomplished by investigating row or column number four.

Course Number

<table>
<thead>
<tr>
<th>Course Number</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td>15</td>
<td>5</td>
<td>11</td>
<td>8</td>
<td>1</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>15</td>
<td>X</td>
<td>19</td>
<td>6</td>
<td>10</td>
<td>11</td>
<td>11</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>19</td>
<td>X</td>
<td>6</td>
<td>4</td>
<td>12</td>
<td>9</td>
<td>15</td>
</tr>
<tr>
<td>4</td>
<td>11</td>
<td>6</td>
<td>6</td>
<td>X</td>
<td>12</td>
<td>1</td>
<td>4</td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>10</td>
<td>4</td>
<td>12</td>
<td>X</td>
<td>9</td>
<td>4</td>
<td>9</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>11</td>
<td>12</td>
<td>1</td>
<td>9</td>
<td>X</td>
<td>12</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>11</td>
<td>9</td>
<td>4</td>
<td>4</td>
<td>12</td>
<td>X</td>
<td>12</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>4</td>
<td>15</td>
<td>12</td>
<td>9</td>
<td>6</td>
<td>12</td>
<td>X</td>
</tr>
</tbody>
</table>

Fig. 1.—The Complete Conflict Matrix

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Defined below is a system of notation which will be useful in the development of RAM.

- \( n \) = the number of single-section courses to be scheduled
- \( p \) = the number of periods in the school day
- \( n/p \) = the number of courses to be scheduled in each period; \( n/p \) must be an integer
- \( N \) = a single-section course
- \( t_N \) = the total conflict of \( N \) with all other courses which have not yet been scheduled
- \( t_{N}^{*} \) = the largest \( t_N \)
- \( C \) = the conflict matrix as shown in Figure 2
- \( C_{ij} \) = the conflict between course \( N_i \) and course \( N_j \)
- \( C_{N,\min} \) = the course having the smallest conflict with course \( N \)
- \( TC \) = the total conflict in the schedule defined by \( S \)
- \( S \) = the solution set containing \( p \) members
- \( M \) = a member of the solution set which defines one combination of courses that will minimize total conflict in the schedule.

RAM is an iterative process. Each of the \( p \) iterations selects one combination of courses to be taught in one period. At the heart of RAM is a set of simple, yet powerful heuristics which enables it to find an optimal or nearly optimal solution set. These heuristics are based on the following considerations:
1. Each course conflicts to a greater or lesser degree with all other courses present in C.

2. A measure of the amount any one course conflicts with all other courses can be obtained by summing the columns of C, i.e., finding $t_N$ for all N.

3. By selecting an M in iteration k, such that the greatest amount of conflict is eliminated for M's in iterations $k + 1$ through $k = p$, TC can be minimized.

The RAM algorithm operates as follows: Choosing an N such that $t_N = t_N^*$, and pairing it with the course with which it has the smallest conflict $(C_N, \text{min})$, we arrive at one member of S. This process is repeated exactly p times until all courses are paired. Obviously, once two courses have been paired, they must be excluded from consideration in future iterations.

Occasionally, problems will be encountered when there are ties for $t_N^*$ or $(C_N, \text{min})$. When these ties are encountered at a decision point, they can be resolved according to the following rules:

1. When there is a tie for $t_N^*$, there is no way to break the tie. In order to insure that a schedule having the smallest total conflict (TC) is found, the problem must be worked through to completion selecting first one and then the other course. Then select the S which has the smallest value of TC.
2. When there is a tie for $C_{N,\min}$ choose the course which has the largest $t_N^*$.

3. When there is a tie for $C_{N,\min}$ and the corresponding $t_N^*$'s are equal, choose either course.

At this point, an example will help to illustrate the way in which RAM is applied to a conflict matrix. The problem defined by Figure 3 will be used where $n = 8$, and assuming that $p = 4$.

**First Iteration ($k = 1$)** Total the columns. Then choose $t_N^*$, i.e., the course having the largest total conflict. In Figure 3, 76 is the largest column total and corresponds to course number two.

Some other course must now be paired with two. This is accomplished by scanning row two and choosing the course with which course two has the least conflict ($C_{N,\min}$). Investigating row two, it is found that course eight fits this condition. The first element of S has now been defined: Teach courses two and eight during the same period.

**Second Iteration ($k = 2$)** Total the columns, excluding elements in rows two and eight and columns two and eight. (The exclusion prevents courses two and eight from entering the solution set in later iterations.) Choose the course with the largest total conflict. This time $t_N^*$ corresponds to course five. Scanning row or column five reveals that courses three and seven both have $C_{N,\min} = 4$. 

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
### Course Number

<table>
<thead>
<tr>
<th>Course Number</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td>15</td>
<td>5</td>
<td>11</td>
<td>8</td>
<td>1</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>15</td>
<td>X</td>
<td>19</td>
<td>6</td>
<td>10</td>
<td>11</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>19</td>
<td>X</td>
<td>6</td>
<td>4</td>
<td>12</td>
<td>9</td>
<td>15</td>
</tr>
<tr>
<td>4</td>
<td>11</td>
<td>6</td>
<td>6</td>
<td>X</td>
<td>12</td>
<td>1</td>
<td>4</td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>10</td>
<td>4</td>
<td>12</td>
<td>X</td>
<td>9</td>
<td>4</td>
<td>9</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>11</td>
<td>12</td>
<td>1</td>
<td>9</td>
<td>X</td>
<td>12</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>11</td>
<td>9</td>
<td>4</td>
<td>4</td>
<td>12</td>
<td>X</td>
<td>12</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>4</td>
<td>15</td>
<td>12</td>
<td>9</td>
<td>6</td>
<td>12</td>
<td>X</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Column totals ($t_N$) for $k$th iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k = 1$</td>
</tr>
<tr>
<td>50</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
</tbody>
</table>

**Fig. 3.**—An illustration of RAM
Referring to rule two for breaking ties above, we find that $t_3 > t_7$ and therefore courses five and three become the second member of $S$.

**Third Iteration ($k = 3$)** Total the columns excluding elements of columns and rows two, eight, three and five. $t_7 = t_{N*}$. Since $C_{7, \text{min}}$ corresponds to course four, four and seven become the third element of $S$. Notice that course five has the same amount of conflict with seven as four. However, five was not considered because it was paired in the previous iteration.

**Fourth Iteration ($k = 4$)** Since there is only one pair of courses which have not been eliminated, they become the last member of $S$. Therefore, one and six become the final member of the solution set.

Summarizing:

$$S = (2, 8/ 3, 5/ 4, 7/ 1, 6)$$

and $\text{TC} = C_{2, 8} + C_{3, 5} + C_{4, 7} + C_{1, 6}$

$$= 4 + 4 + 4 + 1$$

or $\text{TC} = 13$

In the example illustrated, the solution set found by RAM is the optimal solution. No other pairing of courses will yield a schedule with a smaller value of $\text{TC}$. However, it should be repeated that RAM will not find the optimal schedule in all cases. The performance of RAM under various conditions will be discussed in Chapter IV.
Expanding the Usefulness of RAM

Thus far, only the basic principles of RAM have been illustrated. There are five refinements which can be made to the RAM process which will improve its value as a practical technique. The first refinement will enable RAM to handle an odd number of courses.

Suppose that \( n = 7 \) and \( p = 4 \). Since RAM requires that \( n/p \) be an integer, an adjustment is required. The technique employed is similar to that used in the classical assignment problem when \( n \) men are to be assigned to \( n - 1 \) jobs.\(^3\) By adding a "dummy" course, which has zero conflict with all other courses, the integer restriction on \( n/p \) can be satisfied. The implementation requires adding a row and a column of zeros. The column of zeros represents the "dummy" course. RAM will then pair this course with some "real" course, \( N \), for which all \( C_{ij} > 0 \). The interpretation of such a pairing is to teach \( N \) in a period by itself. The technique can also be extended to the case where \( n = 6 \) and \( p = 4 \). Here, two "dummy" courses would be included in \( C \). RAM would pair two "real" courses in two periods, and indicate that the remaining two courses be scheduled for separate periods.

Occasionally, certain course pairs must be excluded from $S$. For instance, one teacher may be assigned to teach two single-section courses, say $N_4$ and $N_7$. Obviously, these two courses cannot meet during the same period. A second adjustment to $C$ will preclude this possibility. To implement this constraint, $C_{4,7}$ and $C_{7,4}$ are assigned a value larger than any other $C_{ij}$. Because RAM chooses $C_{N,\text{min}}$, the pairing 4,7 will never be made. (The assigned value is not included in the column totals, $t_N$.) Another instance in which exclusions are required is the case where two single-section courses must use the same room. German I and Spanish II cannot be scheduled for the language lab during the same period. The use of a large value for the conflict of these two courses will eliminate the possibility of German I and Spanish II being scheduled simultaneously.

The third refinement involves the factor to be assigned each student conflict in the compilation of the conflict matrix. Assigning the factor of one to all conflicts would yield a matrix which counts all conflicts equally. However, by using larger factors for certain students, a system of priorities can be established. For example, it may be desirable to assign a higher priority to the conflicts of seniors. To accomplish this, each senior conflict could be counted as two. It should be noted that any group of factors can be used to simulate a system of
priorities. However, these values should accurately reflect the scheduling priorities established by the school administration.

The fourth refinement provides an indication that all remaining courses have no conflict among them. This occurs when \( t_N = 0 \) for all remaining \( N \). Therefore, the remaining courses can be taught during the same period with no increase in \( TC \). This refinement provides a means to stop the algorithm before \( k = p \), thus saving a number of iterations.

The final adjustment will illustrate the approach to be used when \( n/p > 2 \), i.e., scheduling three or more courses for one period. RAM is based on the concept of eliminating the greatest amount of conflict with each iteration. Since this concept is not dimension-dependent, RAM may be extended to any number of dimensions in order to schedule any desired number of courses for each period. RAM requires that the conflicts between courses to be scheduled together be known. Thus, when scheduling two courses for the same period, a conflict matrix of two dimensions reflects all possible conflicts. However, when three courses are to be scheduled, some potential conflicts are subsets of other conflicts and must be excluded.

Reference to Figures 4a and 4b will help to clarify the point. In Figure 4a, the circles A, B, and C represent three courses. The intersection of the circles
Fig. 4a.—Conflict between three courses.
No student requesting all three.

Fig. 4b.—Conflict between three courses.
One student requesting all three.
represents the number of conflicts between two courses. For example, courses A and B have enrollments of thirty and twenty-eight respectively, with five potential conflicts. If all three courses are scheduled during the same period and no student elects all three, the proper expression for the amount of potential conflict is \((A \cap B) + (A \cap C) + (B \cap C) = 15\). However, if one student elects all three, the proper expression is \((A \cap B) + (A \cap C) + (B \cap C) - (A \cap B \cap C) = 14\). Since a two dimensional matrix cannot reflect the last element of the preceding expression, a three dimensional matrix must be used.

Figure 5 illustrates a three dimensional conflict matrix for the problem of scheduling six courses in two periods. For ease of presentation, one dimension is split into six planes, numbered one through six. Each element of a plane represents the conflict between the course corresponding to the plane number and the two courses corresponding to the row and column number. For example, the potential conflict between one, two, and six can be found in plane one at the intersection of row two and column six. Its value is one. Alternately, the value can be found in plane two at the intersection of row six and column one, etc.

RAM's operations in three dimensions are analogous to those in two dimensions. In order to find the course
Fig. 5.--Three dimensional conflict matrix
having the largest total conflict with all other courses, \( t_N \), planes, rather than columns are totaled. The plane having the largest total \( (t_N^*) \) is selected. The course represented by this plane is scheduled with the two courses which correspond to the minimum element of the plane \( (C_N, \text{min}) \).

An illustration of the application of RAM in three dimensions, is shown in Figure 5. \( t_N \) is displayed in the lower right corner of each plane. \( t_N^* \) is found in the fifth plane. The minimum element of this plane has the value two and corresponds to courses three and six. Therefore, the first member of \( S \) would be: schedule courses five, three and six during the same period. Since there are only three courses remaining they will be scheduled for the other period. If there were three periods and nine courses to be scheduled, however, the process would be iterated three times, with each iteration excluding courses previously scheduled from consideration. Figure 6 represents a flowchart of the RAM process.
Start

k = 0

Construct the Conflict Matrix

Compute all \( t_N \)

k = k + 1

All \( t_N = 0 \) No

Select \( t_N^* \)

For the \( N \) indicated by \( T_N^* \), find \( CN, MIN \)

Schedule remaining courses for one period

Select \( CN, min \) such that \( t_i > t_j \)

Yes

\( Ci, min = C_j, min \)

No

Select all courses scheduled

Yes

No

Exclude courses corresponding to \( CN, min \) from further iterations

End

Fig. 6.—RAM flowchart
A Fortran II program for the computerized resolution of student conflicts by the Rapid Approximation Method for scheduling two courses per period, is shown in Figure 7. The program was written for use on an IBM 1620 computer with 40K core memory. It can accommodate up to forty separate courses and up to eight course requests per student for an unlimited number of students.

The RAM program requires a list of single-section course numbers, punched one per card as shown in Figure 8. This list must include every single-section course to be considered. It may be necessary to prevent the scheduling of certain courses during the same period. If this option is desired, two course numbers are punched on one card. For instance, suppose that courses A, B, and C all require the same classroom. By including course B on input card A, and including course C on card B, courses A, B, and C will not be scheduled for the same period.

After the course number cards are read in, the program requires data for the conflict matrix. The information may be read in using one of two options. The first option reads student request cards. One card is required for each student. On each card the student’s name and any other identifying information is punched in columns one through thirty. Columns thirty-one and thirty-two contain
GLOSSARY

NUMCOR - LIST OF SINGLE-SECTION COURSE NUMBERS
NUM - LIST OF COURSE NUMBERS BROKEN INTO FOUR SINGLE DIGITS
NRQST - STUDENT COURSE REQUESTS BY COURSE NUMBER (AT 80 BECOMES INDEX OF COURSE NUMBERS)
W - CONFLICT WEIGHT
C - CONFLICT MATRIX
TOT - COLUMN TOTALS OF C
MMAX - INDEX OF COURSE HAVING LARGEST COLUMN TOTAL
MIN - INDEX OF COURSE HAVING LEAST CONFLICT WITH MMAX

DIMENSION S(6), KHK(40), C(40,40), NRQST(8), TOT(40)
DIMENSION NUM(41,4), NUMCOR(41,2)

READ COURSE NUMBER LIST
DO 10 N=1,41
READ 1006, (NUM(N,J), J=1,4), NUMCOR(N,2)
NUMCOR(N,1)=NUM(N,1)*1000+NUM(N,2)*100+NUM(N,3)*10
IF(NUMCOR(N,1))10,20,10
10 CONTINUE
20 FI=N
J=FI/2.
FI=FI
FJ=J
FJ=FI-FJ
IF(FJ)30,31,30
30 N=N-1
NACEPT=N
GO TO 40
31 NACEPT=N-1
DO 49 J=1,40
KHK(J)=0
DO 49 I=1,40
C(I,J)=0
49 IF(SENSE SWITCH 1)50,141
READ STUDENT COURSE REQUESTS
DO 50 K=1,99999
READ 1000, (S(JJ), JJ=1,6), W, (NRQST(I), I=1,8)
50 IF(NRQST(I))50,141,58
IF(W)60,59,60
58 W=1.
59

Fig. 7.—RAM Computer Program
CHANGE COURSE NUMBER TO COURSE NUMBER INDEX

60 DO 100 JJ=1,8
   DO 90 I=1,N
      IF(NRQST(JJ))70,110,70
   70 IF(NRQST(JJ)-NUMCOR(I,1))90,80,90
   80 NRQST(JJ)=1
      GO TO 100
   90 CONTINUE
   NRQST(JJ)=0
100 CONTINUE

110 IF(JJ-2)140,120,120

COMPILE CONFLICT MATRIX

120 DO 130 J=1,JJ-2
   DO 130 I=1,JJ-1
      IF(NRQST(J))130,130,128
   128 IF(NRQST(I))130,130,129
   129 C(NRQST(I),NRQST(J))=C(NRQST(I),NRQST(J))+1.*W
      C(NRQST(J),NRQST(I))=C(NRQST(J),NRQST(I))+1.*W
   130 CONTINUE
140 CONTINUE
   GO TO 143

READ ELEMENTS OF CONFLICT MATRIX

141 DO 142 I=1,NACEPT-1
   DO 142 J=I+1,NACEPT
      READ,C(I,J)
   142 C(J,I)=C(I,J)
143 DO 146 I=1,N
      IF(NUMCOR(I,2))144,146,144
   144 DO 146 J=1,N
      IF(NUMCOR(J,1)-NUMCOR(I,2))146,145,146
   145 C(I,J)=-1.
      C(J,I)=-1.
   146 CONTINUE
   IF(SENSWITCH 2)1470,149

TYPE CONFLICT MATRIX

1470 IF(N-20)1474,1474,1471
1471 DO 1472 K=1,4
   1472 TYPE 1007,(NUM(J,K),J=21,N)
   1473 TYPE 1001
   DO 1473 I=1,N
      TYPE 1002,NUMCOR(I,1),(C(I,J),J=21,N)
   M=20
   GO TO 1475
1474 M=N
1475 DO 1476 K=1,4
1476 TYPE 1007,(NUM(J,K),J=1,M)

Fig. 7.--Continued
TYPE 1001
DO 1477 I=1,M
1477 TYPE 1002, NUMCOR(I,1), (C(I,J), J=1,M)
149 TYPE 1009
TOTCFI=0
DO 270 L=1,N/2
T=0
DO 180 J=1,N
TOT(J)=0
IF (KHK(J)) 180, 150, 180
C TOTAL THE COLUMNS OF CONFLICT MATRIX
150 DO 180 I=1,N
IF (KHK(I)) 180, 160, 180
160 IF (C(I,J)) 180, 170, 180
170 TOT(J)=TOT(J)+C(I,J) T=T+TOT(J)
180 CONTINUE
IF (T) 180, 1-280, 1801
180 TYPE 1003, (TOT(J), J=1,N)
182 FI=0
C FIND COURSE HAVING LARGEST COLUMN TOTAL
DO 210 J=1,N
IF (KHK(J)) 210, 190, 210
190 IF (FI-TOT(J)) 200, 200, 210
200 FI=TOT(J) FJ=J
210 CONTINUE
MMAX=FJ FC=99999999 FI=0
C FIND COURSE HAVING LEAST CONFLICT WITH COURSE HAVING LARGEST COLUMN TOTAL
DO 260 J=1,N
IF (KHK(J)) 260, 220, 260
220 IF (C(MMAX,J)) 260, 230, 230
230 IF (FC-C(MMAX,J)) 260, 240, 250
240 IF (TOT(J)-FI) 260, 260, 250
250 FC=C(MMAX,J) FI=TOT(J) FJ=J
260 CONTINUE
MMIN=FJ KHK(MMAX)=1 KHK(MMIN)=1
TOTCFI=TOTCFI+C(MMIN, MMAX)
C TYPE ONE COMBINATION OF RAM SCHEDULE
270 TYPE 1004, NUMCOR(MMAX,1), NUMCOR(MMIN,1), C(MMIN, MMAX)
270 TYPE 1005, TOTCFI
GO TO 283
280 TYPE 1005, TOTCFI
TYPE 1008

Fig. 7.--Continued
DO 282 I=1,N
   IF(NUMCOR(I,1))2801,282,2801
2801 IF(KHK(I))282,281,282
281 TYPE, NUMCOR(I,1)
282 CONTINUE
1000 FORMAT(6A5,F2.0,8(2X14))
1001 FORMAT(I1X)
1002 FORMAT(15,20(I3))
1003 FORMAT(6HTOT(J), 10(2X,15))
1004 FORMAT(2(I4,2X),10HC0NFLICT =,15)
1005 FORMAT(16HTOTAL CONFLICT =F12.0)
1006 FORMAT(411,1X,14)
1007 FORMAT(5X20(I3))
1008 FORMAT(31HC0MBINE FOLLOWING IN ANY MANNER)
1009 FORMAT(///18H** RAM SCHEDULE **///)
283 STOP
END

Fig. 7.--Continued
Fig. 8.—RAM Input Data
a two digit numeric priority to be assigned this student's
conflicts. If these two columns are left blank, a priority
of one is automatically assigned by the program. Columns
thirty-three through eighty contain up to eight four digit
course requests (right justified) separated by two blank
columns. This format is identical to the course request
cards used by the IBM STUDENT assignment program. Thus,
the RAM program can be used conveniently in conjunction
with STUDENT.

If a conflict matrix has been previously compiled,
it can be read directly into the program using the second
input option. The elements of the upper right section of
the matrix are punched one element per card, starting with
the first row of the conflict matrix. The elements of the
first row are punched from left to right, then the second
row and so on. Elements of the main diagonal are not
included.

Two output options are provided by the program.
First, the conflict matrix may be printed or suppressed.
This information will be useful if the matrix has not been
compiled previously. Second, the column totals for any
iteration may be printed, if desired. The input and output
options are controlled by the "sense-switch" setting on
the 1620 computer console.

---

Regardless of the input and output options selected the program will output a list of paired course numbers. Each pair indicates the two courses which should be scheduled during the same period in order to minimize the total number of conflicts for single-section courses. If, during any iteration, the courses remaining to be scheduled have no conflict among them, the program will print the characteristic message, "combine the following in any manner," followed by a list of courses remaining to be scheduled.

The RAM technique was presented in this chapter together with a computer program to speed the computational process. The method is based on heuristic decision rules which tend to minimize student conflict in single-section courses. RAM does not, however, generate an optimal schedule in all cases. In an attempt to illustrate RAM's ability to find an optimal schedule evaluation tests are presented in Chapter IV. Additionally, the method is tested as a practical tool for scheduling classes at two schools.
CHAPTER IV

TESTING THE METHOD

Several tests conducted on the RAM techniques introduced in the previous chapter will be discussed in this chapter. The first section is devoted to practical applications of RAM. In the second section the ability of RAM to find an absolute minimum conflict schedule is investigated.

Two Practical Tests

Two practical applications of RAM were attempted. The first test of RAM's ability to schedule courses was conducted on data for the University of Montana MBA program at Malmstrom Air Force Base. The second test was conducted on data for Great Falls High School, Great Falls, Montana. These two schools were selected to illustrate the wide range of scheduling problems which RAM can accommodate.

The algorithm was first applied to the type of student scheduling problem which RAM was originally intended to solve. The RAM computer program was used to schedule courses for the Spring Quarter of 1970, at Malmstrom Air Force Base.
Before the quarter began, the school's administrator selected eight single-section courses which were to be taught during the four period day.

Each course number was punched on a single card as shown in Figure 8. Because one professor was scheduled to teach two courses, 512 and 562, these courses could not be scheduled during the same period. Therefore, a second course number, 562, was punched on the course card for 512. Similarly, course 680 was added to the 690 course number card. Schedule combinations which are inadmissible appear in the conflict matrix of Figure 8 as -1.

After the list of available courses was established, each of the one hundred and ten students selected up to four courses in which he wished to enroll. Then one student request card was punched for each student. Because the school has not established a conflict priority, columns 31 and 32 were left blank.

The RAM program together with the input data were read into an IBM 1620 computer. The computer output the conflict matrix. After applying the RAM algorithm, the 1620 also output the proposed schedule for the Spring Quarter shown in Appendix B.

The schedule shows that courses 692 and 543 should be taught during the same period. Only one student will have a conflict between these two courses. The second line shows that scheduling 680 and 512 together results in no conflicts.
The schedule proposed by the administration of the school would have resulted in six conflicts, while the RAM schedule indicated only two conflicts. Therefore the RAM schedule of courses was adapted by the school for the Spring Quarter.

The second practical application of RAM was made on data obtained from Great Falls High School for the Spring Semester of 1970. This test was conducted to illustrate RAM's ability to schedule a large number of courses.

Great Falls High School had planned to teach forty-seven single-section courses during a nine period day. Since the number of courses which the RAM program can accommodate is limited by the amount of computer memory available, thirty-six single-section courses were chosen at random. The course numbers were read into memory. No restrictions were placed on courses which could meet during the same period because the school does not assign teachers to a section until the schedule of classes is nearly finalized.

Great Falls High uses the IBM STUDENT\(^1\) assignment program which outputs a conflict matrix. Because these data had previously been compiled by STUDENT, the conflict matrix was read directly into the RAM program using the optional

\[^{1}\text{Op. cit., p. 6.}\]
input method. The RAM output for the high school is shown in Appendix C.

The RAM program which was applied to the data for Great Falls High would not normally produce a satisfactory schedule. The thirty-six courses would be scheduled two courses per period yielding an eighteen period day. However because of the number of zero elements in the conflict matrix, the courses can be scheduled into a nine period day after modifying the schedule proposed by RAM.

The program scheduled eleven pairs of courses for different periods and indicated that the remaining fourteen courses could be taught during one period. This schedule has no conflicts and requires twelve periods. The number of periods can be reduced to nine by further grouping course pairs which have no conflict. For instance, by scheduling 527, 29, 213 and 305 in the same period, the number of periods required can be reduced to eleven. By continuing this process the schedule can be reduced to nine periods. The remaining reductions are: schedule 711, 31, 327 and 17 for one period, and 617, 115, 217 and 13 for another period. The resulting schedule will have nine periods and no student conflicts.

The fourteen courses which were not paired can be manually scheduled with paired courses so that the total conflict will not be increased. This will result in a
better balance in the number of courses assigned to one period. Thus, 107 could be scheduled with 19 and 715.

The above procedure for modifying the RAM proposed schedule was included to illustrate that a satisfactory schedule can be achieved when only limited computer memory is available.

The entire program including card input and console typewriter output required only fifteen minutes of 1620 computer time.

Four Hypothetical Tests

The RAM algorithm was designed to find a schedule with the minimum total conflict. The decision rules incorporated in RAM insure that the probability of finding such a schedule is large. In some cases, however, only a near minimal conflict schedule is achieved.

In order to determine how well RAM fulfills its purpose, a computer program was written to evaluate RAM according to the following procedure. First, a random conflict matrix is generated for eight courses. Next, the program investigates every possible schedule which can be made from this matrix. It checks the total number of conflicts in each schedule and selects the smallest, $T_{C_S}$, and the largest, $T_{C_L}$, values. Then the program applies the RAM algorithm to the same matrix and determines the total number of conflicts, $T_{C_{RAM}}$, in the RAM schedule. Using
these values, a measure of how well RAM finds the schedule having the minimum number of total conflicts is calculated according to the following equation:

\[
M = \frac{TC_{RAM} - TC_S}{TC_L - TC_S}
\]

If RAM finds a schedule having the fewest conflicts, then \(M = 0\). If RAM does not find a schedule containing the smallest number of conflicts, then \(0 < M \leq 1\). The smaller the value of \(M\), the closer RAM is to finding a schedule containing the fewest possible conflicts.

After the program evaluates RAM's performance for one matrix, the value of \(M\) is printed and a different matrix is generated. This process is repeated for one hundred matrices. The mean value of all \(M\)'s is then printed.

There are various distributions of the elements of the conflict matrix which can occur. Some schools such as Great Falls High may have a conflict matrix containing a large number of zero elements. In other schools, the elements may appear to be randomly distributed. Figures 9 through 12 show various distributions of the elements of the conflict matrix. The corresponding distribution of \(M\) is given at the bottom of each page.

Figure 9 illustrates the distribution of \(C_{ij}\) for Great Falls High. When this distribution was simulated one hundred times, RAM found a schedule with the minimum number of conflicts every time.
Distribution of Conflicts \((C_{i,j})\)

Mean value of \(M = .000\)

Fig. 9.--Evaluation of RAM
A distribution containing fewer zero elements in the conflict matrix is shown in Figure 10. The range of values which $C_{ij}$ can assume is also larger. The lower graph shows the distribution of $M$. In this case, RAM found a schedule with the fewest number of conflicts ($M = 0$) eighty-eight times out of one hundred. Nine RAM schedules had an $M$ value between zero and .05. The remaining three schedules investigated showed an $M$ value between .05 and .10. The mean value of $M$ for this distribution of $C_{ij}$ was .005.

The result of evaluating RAM when thirty percent of the $C_{ij}$ values are zero is illustrated in Figure 11. Although the distribution of $M$ ranges up to .20, RAM still found a schedule having the fewest possible number of total conflicts in eighty-three cases.

The application of RAM using a conflict matrix in which the values of all elements of $C$ are randomly distributed on the interval 0 to 19 is presented in Figure 12. In fifty-five cases, RAM found a schedule having the smallest number of conflicts.

A chi-square test for goodness of fit was performed to compare the RAM generated $M$ distributions to those that could be expected from choosing a random schedule. In order to obtain an expected distribution of $M$, 5250 schedule combinations were generated for each distribution of $C_{ij}$. These distributions were compared with the RAM generated $M$. 

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
distributions using the chi-square test. In every case the RAM distribution was superior with a .5 percent level of significance.
Distribution of Conflicts ($C_{i,j}$)

Freq. of $C_{i,j}$

Distribution of $M$

Freq. of $M$

Mean value of $M = .005$

Fig. 10.—Evaluation of RAM
Distribution of Conflicts ($C_{i,j}$)

Distribution of $M$

Mean value of $M = .011$

Fig. 11. -- Evaluation of RAM
Distribution of Conflicts $(C_{i,j})$

Distribution of $M$

Mean value of $M = .042$

Fig. 12.—Evaluation of RAM
CHAPTER V

SUMMARY

This final chapter is divided into four sections. The first summarizes the results of tests conducted on RAM. The second section lays out some limitations of the method. The conclusions reached in this research are presented in the third section. Finally, recommendations for further research are proposed in the fourth section.

Results of Testing RAM

The first practical test of RAM was conducted on data of the University of Montana MBA program at Malmstrom Air Force Base during the Spring Quarter of 1970. The administrator had previously established a schedule for classes based on prerequisites. Three courses offered that semester were a prerequisite series. The first course was a prerequisite for the second, and the second a prerequisite for the third. Since no student could elect more than one of these courses, there could be no conflicts between any combination of them. Therefore, the administrator proposed to schedule these three courses during the same period. This decision would have resulted in no conflicts for at
least one period. However, because the courses remaining to be scheduled had a large number of conflicts among them, the minimum number of conflicts that could be obtained by scheduling the prerequisites together was six. In comparison, RAM generated an acceptable schedule which contained only two conflicts.

As a result of RAM's ability to find a better schedule for the Spring Quarter, it has been used each succeeding quarter, by the school.

The second practical test of RAM was performed on data from Great Falls High School. The purpose of this test was to illustrate RAM's ability to handle large scheduling problems. In this respect the RAM computer program did not produce a satisfactory schedule. The computer memory available was not large enough to schedule all forty-seven single-section courses.

This test did illustrate, however, that when a large number of elements of the conflict matrix are zero, an acceptable schedule can be obtained by manual manipulation of the schedule generated by RAM.

A final series of tests was conducted to determine how well RAM generated a schedule containing a minimum number of conflicts. These tests indicated that the number of schedules generated containing an absolute minimum number of conflicts, is proportional to the number of zero elements in the conflict matrix. This is expected, because if all
elements of $C$ were zero, RAM would always produce a schedule containing zero conflict, i.e., the absolute minimum number of conflicts.

**Limitations of the RAM Method**

Although RAM has proved to be a useful tool for the MBA program, there are certain limitations which must be understood before it can be used successfully.

The RAM technique for scheduling can be used only for single-section courses. The conflict matrix is generated with the assumption that a conflict will result if two courses are scheduled during the same period. If they are not scheduled during the same period no conflict will result. However, multiple-sections may or may not meet during the same period and the exact number of conflicts cannot be determined. Since RAM depends on the conflict matrix to choose a schedule, the elements of $C$ must be defined exactly so that a schedule with the least number of conflicts can be generated. Therefore, multiple-section courses cannot be scheduled by RAM.

RAM has a second limitation. It can schedule only an equal number of courses during each period of the day. With the exception of scheduling a "dummy" course, if two courses are scheduled for one period, then all periods must have two courses. If other than an equal number of courses is desired during a period, the scheduler must employ a
manual method similar to the one used to combine courses for Great Falls High School.

From a practical point of view, this restriction is not significant. In most school scheduling problems, there are an equal number of teachers and classrooms available each period. Therefore, most schedules tend to be balanced among the periods of the day.

The third limitation detracts somewhat from the usefulness of the method. RAM can be used only where the length of the schedule cycle is one day, i.e., where the schedule is the same each day of the week.

Conclusions

RAM is not designed to generate an entire master schedule of classes for all school scheduling problems. It can be a useful technique for scheduling an equal number of single-section courses during each period of the school day where the schedule cycle length is one day, and some single-sections must meet during the same period.

RAM does find a schedule containing a minimal number of conflicts in a majority of cases. In instances where the minimal is not found, the probability of finding a schedule with a near minimal number of conflicts is high.

RAM can also be used to exclude certain scheduling combinations from consideration. When one instructor must
teach two single-section courses, these two courses will not be scheduled during the same period.

The computer program presented in Chapter IV can accommodate up to forty courses and an unlimited number of students. The number of courses which can be scheduled is limited by the amount of computer memory available. Computers with larger memories can be programmed to execute RAM on a much larger number of courses. Additionally, RAM uses the same student request cards as STUDENT, making RAM compatible with the STUDENT assignment program.

Recommendations for Further Research

One of the drawbacks of RAM is the limited schedule cycle which can be accommodated by the method. By extending the cycle length, a wider variety of scheduling problems can be attacked. One approach proposed but not investigated in this research would consider the week as a continuous series of periods. By considering a class that must meet twice per week as two separate courses, and by not allowing the separate courses to meet during the same period, it may be possible to extend the cycle length to a week and beyond.

RAM is limited to scheduling single-section courses as explained earlier in this chapter. Although RAM by itself cannot be used to schedule multiple-section courses, it might be used in conjunction with other techniques for the
scheduling of multiple-sections. One such combination would use RAM to establish a core schedule of single-section courses. Then Boyles'\textsuperscript{1} technique could be used to complete the scheduling of multiple-sections.

\footnote{N. L. Boyles, \textit{op. cit.}}
APPENDIX A

Derivation of $C_d = \frac{n!}{(n/p)!p(p)!}$

Let $n =$ number of courses to be scheduled
$p =$ number of periods in the school day
$n/p = r =$ number of courses to be scheduled
per period, where $r$ is a positive integer.

We wish to find the number of distinct combinations of courses, $C_d$, when $r$ courses are selected from a continually decreasing number of courses. For instance when two courses are selected from eight, only six remain from which we must again choose two, leaving four from which we again choose two, etc. We also wish to eliminate like sets of combinations, e.g., $(1, 2/3, 4) = (3, 4/1, 2)$.

In the first period there are $C^r_\eta$ ways of choosing $r$ courses from a set of $n$ elements

$$C^r_\eta = \frac{n!}{r!(n-r)!}$$

In the second period only $n-r$ courses remain from which $r$ must again be chosen

$$C^{n-r}_r = \frac{(n-r)!}{r!((n-r)-r)!} = \frac{(n-r)!}{r!(n-2r)!}$$
In the third period 2r courses have already been eliminated leaving n-2r from which to choose r.

\[ C_{n}^{n-2r} = \frac{(n-2r)!}{r!((n-2r)-r)!} = \frac{(n-2r)!}{r!(n-3r)!} \]

By induction, in the kth period the expression becomes

\[ C_{n}^{n-(k-1)r} = \frac{(n-(k-1)r)!}{r!(n-kr)!} \]

For the last period when k = p

\[ C_{n}^{n-(p-1)r} = \frac{(n-(p-1)r)!}{r!(n-pr)!} \]

Using the multiplication principle, the total number of combinations, C, for p periods is given by

\[ C = \frac{n!(n-r)!(n-2r)!\ldots(n-(p-2)r)!(n-(p-1)r)!}{r!(n-4)!r!(n-2r)!r!(n-3r)!\ldots r!(n-(p-1)r)!r!(n-pr)!} \]

Simplifying, we obtain

\[ C = \frac{n!}{(r!)^p(n-pr)!} \]

but \( r = \frac{n}{p} \)

and \( (n-pr)! = (n-p(n/p))! = (n-n)! = 0! \equiv 1 \)

Therefore, \( C = \frac{n!}{((n/p)!)^p} \)

Furthermore, there are \( p! \) ways in which the combinations for each period can appear.
Therefore, again using the multiplication principle, we obtain the number of distinct combinations

\[ C_d = \frac{n!}{((n/p)!)^p(p)!} \]
APPENDIX B

RAM OUTPUT FOR
MALMSTROM AIR FORCE BASE

\[
\begin{array}{cccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
5 & 5 & 6 & 5 & 6 & 6 & 6 & 6 \\
4 & 1 & 4 & 6 & 5 & 8 & 9 & 9 \\
3 & 2 & 6 & 2 & 0 & 0 & 0 & 2 \\
543 & -1 & 1 & 0 & 7 & 3 & 5 & 2 & 1 \\
512 & 1 & -1 & 0 & -1 & 0 & 0 & 1 & 3 \\
646 & 0 & 0 & -1 & 0 & 0 & 6 & 7 & 4 \\
562 & 7 & -1 & 0 & -1 & 3 & 2 & 1 & 2 \\
650 & 3 & 0 & 0 & 3 & -1 & 9 & 2 & 6 \\
680 & 5 & 0 & 6 & 2 & 9 & -1 & -1 & 6 \\
690 & 2 & 1 & 7 & 1 & 2 & -1 & -1 & 9 \\
692 & 1 & 3 & 4 & 2 & 6 & 6 & 9 & -1 \\
\end{array}
\]

** RAM SCHEDULE **

\[
\begin{align*}
692 & \quad 543 \quad \text{CONFLICT} = 1 \\
680 & \quad 512 \quad \text{CONFLICT} = 0 \\
690 & \quad 562 \quad \text{CONFLICT} = 1 \\
\text{TOTAL CONFLICT} = 2. \\
\text{COMBINE FOLLOWING IN ANY MANNER} \\
646 \\
650 \\
\end{align*}
\]
** APPENDIX C **

** RAM OUTPUT FOR **

GREAT FALLS HIGH SCHOOL

** RAM SCHEDULE **

<table>
<thead>
<tr>
<th>Time</th>
<th>Activity</th>
<th>Conflict</th>
</tr>
</thead>
<tbody>
<tr>
<td>19</td>
<td>715</td>
<td>0</td>
</tr>
<tr>
<td>276</td>
<td>254</td>
<td>0</td>
</tr>
<tr>
<td>25</td>
<td>121</td>
<td>0</td>
</tr>
<tr>
<td>705</td>
<td>117</td>
<td>0</td>
</tr>
<tr>
<td>309</td>
<td>125</td>
<td>0</td>
</tr>
<tr>
<td>527</td>
<td>29</td>
<td>0</td>
</tr>
<tr>
<td>213</td>
<td>305</td>
<td>0</td>
</tr>
<tr>
<td>711</td>
<td>31</td>
<td>0</td>
</tr>
<tr>
<td>617</td>
<td>115</td>
<td>0</td>
</tr>
<tr>
<td>327</td>
<td>17</td>
<td>0</td>
</tr>
<tr>
<td>217</td>
<td>33</td>
<td>0</td>
</tr>
</tbody>
</table>

TOTAL CONFLICT = 0

COMBINE FOLLOWING IN ANY MANNER:

107
111
123
133
135
137
139
141
149
307
315
317
319
611
SOURCES CONSULTED

BOOKS


DISSERTATIONS


ARTICLES


MISCELLANEOUS


Student Scheduling System, Application Reference, NCR Company, Dayton, Ohio.
Student Scheduling System/360, Application Description, IBM Corporation, 1966.

Student, Contributed Program Library #1620-10.3.017, IBM Corporation, 1964.