The purpose of this study is to determine if the problem of scheduling instructors to classes can be formulated as a linear program involving integer variables limited to values of zero and one. Once such a formulation is accomplished, attempts at electronic computer solution of the problem are considered.

Specifically, the study concerns itself with the scheduling of Oregon State University Chemistry Department graduate assistants to teach laboratory sections. An algorithm proposed by Egon Balas for solving linear programs with zero-one variables is discussed and applied to the example.
AN INTEGER PROGRAMMING PROCEDURE
FOR SCHEDULING INSTRUCTORS

by

THOMAS LEYBA YATES

A THESIS
submitted to
OREGON STATE UNIVERSITY

in partial fulfillment of
the requirements for the
degree of

MASTER OF SCIENCE

June 1967
APPROVED:

Redacted for privacy

Professor of Statistics
In Charge of Major

Redacted for privacy

Chairman of Department of Statistics

Redacted for privacy

Dean of Graduate School

Date thesis is presented  August 3, 1966

Typed by Jan Lewis
ACKNOWLEDGMENT

The author wishes to express his appreciation to Dr. Donald Guthrie for the encouragement and guidance he provided during the preparation of this thesis.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>II</td>
<td>UNIVERSITY SCHEDULING PROBLEMS</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>General Discussion</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Chemistry Department Laboratory</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Instructor Assignments</td>
<td>8</td>
</tr>
<tr>
<td>III</td>
<td>LINEAR PROGRAMMING</td>
<td>11</td>
</tr>
<tr>
<td>IV</td>
<td>BALAS' ALGORITHM FOR SOLVING LINEAR PROGRAMS WITH ZERO-ONE VARIABLES</td>
<td>18</td>
</tr>
<tr>
<td>V</td>
<td>FORMULATION OF THE SCHEDULING OF CHEMISTRY DEPARTMENT GRADUATE</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>ASSISTANTS AS A LINEAR PROGRAM</td>
<td></td>
</tr>
<tr>
<td>VI</td>
<td>CONCLUSIONS</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>BIBLIOGRAPHY</td>
<td>28</td>
</tr>
</tbody>
</table>
A scheduling procedure for assignment of facilities using integer linear programming techniques is developed in this paper. The procedure used will lead to an optimum solution based on a least cost criterion.

A general discussion of university scheduling problems is presented, leading to a statement of a specific problem: the scheduling of graduate assistants to teach laboratory sections in the Chemistry Department. This is followed by a brief discussion of linear programming and more specifically linear programming problems which involve variables which take on only the values zero and one. An algorithm for solving such problems is introduced and, finally, the graduate assistant assignment problem is formulated in terms of the algorithm, and conclusions drawn from tests run on an electronic digital computer are presented.
CHAPTER II

UNIVERSITY SCHEDULING PROBLEMS

General Discussion

University scheduling problems have attracted the interest of mathematicians, statisticians and computer scientists for the past decade. Administrators have shared this interest, since the scheduling problems have generally involved manipulation of large, rather complex files of information. These files include information in regard to courses, sections of courses, meeting times, available facilities, student course requests, etc.

Scheduling problems have generally fallen into five distinct categories:

(1) Course meeting time master schedule
(2) Student sectioning
(3) Room (facilities) assignment
(4) Instructor assignment
(5) Final examination schedule

Systems and procedures which have been developed in the past to handle one or a combination of these categories have pointed up the fact that the efficiency of the system is largely a function of factors not always recognized by the researcher. Aside from the need
for an efficient algorithm, the system's speed and accuracy are affected by

(a) **Flexibility of administrative policy.** Exceptions create complications in machine processing. If the assignment of facilities is to be based on a combination of size of room, blackboard space, distance from instructor's office, etc., the assignment is fairly straightforward. However, if some departments are allowed to specify that certain of these factors have no priority (for instance, size of room) then the machine procedures become very difficult to define.

(b) **Data collection procedure.** Editing frequently constitutes a major portion of the system. Data collected in machine readable form (such as mark sensed cards) or data directly entered into a computer via remote terminals may constitute a substantial savings over traditional coding-card punching-verifying operations.

(c) **Size of problem.** The method for using computers to solve a problem for a small institution may be grossly inefficient on larger problems. Generation of a new schedule of classes for a small institution may be done with the necessary files kept entirely in the memory of the computer, while the same problem for a large institution would
require use of peripheral storage devices such as magnetic tape or disk.

The most popular of the scheduling problems has been student sectioning. James F. Blakesley (4, 5) of Purdue is generally considered to be the pioneer leader in this area with PASS (Purdue Academic Scheduling System) (5) being the culmination of development work over an eight year period and several computer systems. Systems such as PASS are in operation throughout the year, many weeks of work being required to build the data files for the class schedules, room assignments, instructor assignments, etc. After the files are organized, the student requests are processed through editing routines to confirm their legitimacy prior to the actual scheduling run. These edit runs can also be used to provide warning of unusually heavy or light demands for certain courses. The sectioning runs then follow, with fee statements and schedules resulting.

Other institutions have used the Purdue student sectioning system as a model. The University of Massachusetts, in cooperation with International Business Machines Corporation (15), has an IBM 7090 system which will accommodate up to 1000 courses with 2500 sections and will schedule 5000 students per hour. Washington State University (8) has implemented a similar system on an IBM 709, with work presently being carried out to convert the programs
to the IBM 360. This is the only computerized student sectioning system in the western part of the United States. Massachusetts Institute of Technology has installed a similar system on their Control Data Corporation 3600 computer.

Akin of International Business Machines Corporation has developed a system, entitled CLASS (1), which has been used by approximately 100 secondary schools on a service bureau basis. The CLASS system is limited to 300 courses with 900 sections total. It is estimated to operate at a speed of 60 to 200 students per minute on the IBM 7070 computer.

A similar service bureau approach to scheduling secondary schools has been developed at the Educational Facilities Laboratories at Massachusetts Institute of Technology with a system entitled GASP (Generalized Academic Simulation Programs) (18). However, both GASP and a similar system developed at Stanford University (S: Stanford School Scheduling System) are "complete" systems in that they also generate class master schedules and facilities assignments.

Indiana University (13) has taken a somewhat different approach to the student scheduling problem. The systems cited above all operate with batches of requests. Thus student requests are all entered into the system at the same time. The Indiana system is based on remote terminal entry of individual requests with immediate scheduling or notification that scheduling is impossible and for what
reason. The Indiana students have an option by which they may communicate with a remote terminal operator by telephone to register their requests and obtain confirmation of schedule. Also, Indiana still permits students to select specific sections of multi-sectioned courses, an option not included in the other systems.

Computerized student scheduling systems have also been developed at smaller colleges on relatively small computer systems. Examples include an IBM 1620 program in use at Western Carolina College (16), and a CDC 160-A program developed and used at College of St. Thomas (21).

Final examination scheduling and the room assignment problem have both received considerable attention. Cornell University has utilized a CDC 1604 computer for construction of final exam schedules, and J. E. L. Peck and M. R. Williams of the University of Alberta (Calgary) (19) have developed a generalized procedure for exam scheduling which they have implemented on an IBM 1620 computer. A project currently underway at Oregon State University plans to make room assignments to individual sections in such a way that the facilities will be used in an optimum manner.

As mentioned previously, master schedule generation by computer is a reality on the secondary school level using systems such as GASP and $S^4$. A procedure for scheduling mathematics lectures at Queen Mary College, London, England, also has been published
There are no references available to indicate a successful solution to generation of an entire university master schedule by computer. Blakesley and associates at Purdue are in the process of testing CUSS (Comprehensive University Scheduling System) which, when completed, will provide a dynamic scheduling tool which will make assignments of meeting times to classes and of classes to rooms. The present version of CUSS being tested on an IBM 7094 has a capacity for 400 courses and 200 rooms.

The author has also conducted a study of the feasibility of generating master schedules by computer. This work was reported to the Oregon State System of Higher Education Inter-institutional Committee on Computer Activities, which authorized the study.

No record has been found of earlier research into the problem of assigning instructors to classes. This is not too surprising when one considers the fact that this problem is one rather intimately involved with personalities and other non-quantitative factors. Further, the assignment of an individual has generally been considered a permanent appointment (carrying over from year to year), not necessarily subject to change.

However, there is ample justification for considering this category of scheduling problems. The most obvious application involves the assignment of a large number of individuals to sections
of the same or similar courses. Another possibility would be to use such a procedure as a first estimate upon which a department chairman could make necessary changes to take into account the human factors cited previously.

Chemistry Department Laboratory Instructor Assignments

An example of a rather large instructor assignment problem is the assignment of graduate assistants to teach laboratory and recitation sections in the Oregon State University Chemistry Department. Each term during the regular school year approximately 60 graduate assistants are given assignments to teach in some 50 laboratory and recitation sections. Naturally, many of the assistants must take course work, so the scheduling problem involves the intersection of two schedules, that of the laboratories and that of the graduate assistants' courses. Laboratories are scheduled Monday through Friday, 8 a.m. to 5 p.m. and Saturday mornings, so there are over 50 hours per week during which conflicts may occur. Besides the problem of time conflicts (which may also involve activities other than classes, such as staff meetings, outside employment, etc.), the assignment must take into account the field of specialization of the graduate assistant, since it is not desirable to have a Biochemistry major teaching Physical Chemistry laboratories. To minimize the number of staff meetings and to simplify the laboratory
preparation for the assistants, the assignments for each individual should be entirely within the same course. As an example, an individual usually would not be assigned laboratories from both Ch 101 and Ch 201.

Other constraints that must be considered include the work load associated with each laboratory section, the work load upper bound for each individual graduate assistant, the number of assistants required for each laboratory section and the sets of laboratory sections that have concurrent meeting times.

If the object of the procedure is to optimize the assignment (i.e., minimize inefficiencies), then "costs" must be assigned to every potential graduate assistant-laboratory section combination. These costs must reflect assistant-laboratory schedule conflicts, individual graduate assistant requests for assignment, departmental preferences for assignments, etc.

Assignments must be made for over a dozen different courses. However, except for the General Chemistry courses (the Ch 101-2-3, Ch 201-2-3 and Ch 104-5-6 sequences), the assignments do not involve multiple sections and the "best assignment" is usually a matter of administrative judgment based on the background of the graduate assistants. Thus, the problem to be dealt with in this study has been restricted to 45 graduate assistants and 117 sections in five General Chemistry courses (Ch 101, 103, 201, 204, and 204H) for fall term 1965.
The task of making these assignments has required from two to four working days for a senior member of the Chemistry Department staff.

This chapter has been an exposition of the university scheduling problem area, with special attention given to the assignment of graduate assistants to laboratory sections in the Chemistry Department at Oregon State University. Chapter III will introduce linear programming techniques which will lead to a procedure for handling the assignment problem.
CHAPTER III

LINEAR PROGRAMMING

Linear programming is a mathematical technique by which one can determine an optimum allocation of resources which will obtain an objective (such as minimum cost or maximum return) when there are alternative uses for the resources.

Hadley (11, p. 20-21) notes in discussing the history of linear programming that such problems first arose in economics, with the early development of the method taking place in the 1930's. However, as Hadley explains, it was the work of George Dantzig and associates on World War II Air Force research into allocation of resources which led to the formulation of the generalized linear programming problem. Dantzig devised the simplex method of solution in 1947, but his work was not generally available until 1951.

Hadley's definition of linear programming is sufficient for our purposes, to wit, "Linear programming deals with that class of programming problems for which all relations among the variables are linear. The relations must be linear both in the constraints and in the function to be optimized" (11, p. 2), and, "Given a set of m linear inequalities or equations in r variables, we wish to find non-negative values of these variables which will satisfy the constraints
and maximize or minimize some linear function of the variables" (11, p. 4).

Thus, provided that we can use linear relationships to describe both the objective and the constraints of the system under study, we wish to impose a procedure which will lead to an optimum solution of the problem. More specifically, a problem to be solved by linear programming must have:

(1) a definite, identified, numerical goal. This "objective function" must be stated as the sum of a series of terms, each of which is the product of an activity level and the unit cost associated with that activity. This is the function to be minimized or maximized.

(2) activities that are measurable in numeric terms. These activities must be individually identifiable.

(3) linear relationships among sets of the activities which can be stated as inequalities or equations. These are the constraints.

Linear programming includes the formulation of the problem as well as the solution (i.e., finding the optimum policy). Failure of the computing procedure to find a solution frequently is due to inaccurate formulation of the problem. Most linear programming problems, if large enough that a solution cannot be deduced intuitively, require the application of an electronic computer. Generalized
programs for the solution of linear programming problems involving continuous valued variables are available for most computer models.

Before further describing linear programming, it is necessary to introduce the following definitions:

**Activity:** A structural variable \( x_i \) whose level is to be computed in a programming problem.

**Activity Level:** The value taken on by a variable in a solution to a programming problem.

**A-matrix:** The rectangular array of constraint coefficients with \( m \) rows and \( r \) columns, where \( m \) is the number of constraints and \( r \) the number of activities.

**Basic Feasible Solution:** A basic solution to the constraint equations in which the non-zero values of the variables meet all of the bound restrictions. The lowest cost basic feasible solution is the optimum.

**Basic Solution:** A solution to the row constraint equations in which the number of variables that assume non-zero values does not exceed the number of equations.

**Cost Vector:** The row of objective function coefficients of a cost minimization problem.

**Feasible Solution:** A solution to the constraint equations in which all variables satisfy their sign restrictions.

**Objection Function:** That function of the independent variables whose maximum or minimum is sought in an optimization problem.
**Requirement Vector**: The column of constant values appearing on the right hand side of the constraint equations. Also referred to as the b-vector.

**Simplex Method**: A computational routine for obtaining the optimal solution to a linear programming problem. It is an iterative elimination procedure at each stage yielding a basic solution, and it rests primarily on the following two principles:

1. Elementary row operations on the constraint matrix leave the set of feasible solutions unchanged.

2. The number of non-zero values in an optimal solution is never more than the number of constraint equations.

The following steps comprise one iterative stage:

a. A test of whether the current solution is optimal and/or feasible.

b. If not both optimal and feasible, a choice of an entering variable and a departing variable.

c. A pivot step so as to read off the new solution: then back to a.

**Slack Variable**: An auxiliary variable \( y_i \) introduced to convert an inequality constraint to an equation.

**Solution**: A set of values of the variable which satisfy the given constraints.
The general linear programming problem then can be stated as:

Given \( Ax \leq b \)

minimize (or maximize) \( c^T x = z \)

subject to all \( x \geq 0 \)

The simplex method (7, 11, 17), and variations thereof, will compute basic feasible solutions (feasible solutions which are also basic solutions) in an orderly fashion such that the value of \( z \) is monotonically decreasing (or increasing) until no such change in \( z \) is possible. The last basic feasible solution encountered will be the optimum solution. This is true even in the case where the optimum solution is unbounded, which, as noted earlier, is usually due to errors in problem formulation.

The solution of linear programming problems in which part or all of the variables have integer variables require special techniques. The "common sense" approach of handling such problems as normal linear programming cases and rounding the results to the nearest integer frequently fails to produce an optimum answer. Hadley (12, p. 251-258) treats these integer problems as a special case of non-linear programming. Dantzig (7, p. 514-550) and Llewellyn (17, p. 264-273) include integer programming as "extensions" of linear programming.

Examples of integer programming problems include capital
investment planning, the allocation of machines or men to projects, travel route planning to minimize the distance covered in visiting 'n' locations, and the classical "Knapsack Problem." Thornton (20) has investigated the use of integer programming on an assignment problem involving the scheduling of aircraft for Australian domestic airlines.

Early work in the area of integer programming problems by Dantzig (7, p. 514-520) and others failed to develop a generalized procedure for their solution. Finally, in 1958, Gomory (10) proposed a method which calls for solution of the problem by the simplex method using real variables, at which point additional constraints are developed which impose discreteness on those variables which require it. The new constraints are known as "cutting-plane constraints." Gomory proves that the number of additional constraints cannot exceed the number of original variables plus one. Unfortunately, for even moderately large problems, this is a serious additional requirement on computer memory resources.

An even more restrictive linear programming case involves the class of problems whose variables take on only the values zero or one. Dantzig (7, p. 536-538) discusses these dichotomous problems, and Gomory's algorithms are applicable to this case. Balas (3) has developed an additive algorithm for solving the zero-one variable linear programming problems, and it is this procedure
which will be used to seek a solution to the chemistry laboratory assignment problem.

This chapter has developed the background of linear programming preparatory to a more detailed examination in Chapter IV of an algorithm for solving problems that can be classed as linear programming and whose variables are restricted in value to zero or one.
CHAPTER IV

BALAS' ALGORITHM FOR SOLVING LINEAR PROGRAMS
WITH ZERO-ONE VARIABLES

Balas' algorithm deals with the problem stated as:

Find \( x \) such that

\[
z = c^T x \quad \text{minimum}
\]  

Subject to \( Ax + y = b \)

\[
x_j = 0 \text{ or } 1
\]

\[
y \geq 0
\]

Where \( c_j \geq 0 \).

\( A \) is an \( m \times n \) matrix.

An \( (n + m) \)-dimensional vector \( u(x, y) \) will be called a solution if it satisfies (2) and (3);

a feasible solution if it satisfies (2), (3) and (4); and an optimal feasible solution if it satisfies (1), (2), (3) and (4).

Note that the constraints are stated entirely as inequalities of the form \( \leq \) and the costs are all \( \geq 0 \). The procedure for converting a problem not already in this form is:

1. Replace all equations by two inequalities.
2. Multiply all inequalities of the form \( \geq \) by \(-1\).
3. Convert all \( x_j = 1 - x'_j \) for \( c'_j < 0 \), where \( x'_j \).
and $c_j'$ from original statement of the problem.

Important computational features of Balas' algorithm are:

It is completely additive in execution; the coefficient matrix $A$ is never modified; the problem is directly attacked as a zero-one integer problem rather than first requiring the solution of the ordinary integer program.

The general problem as stated has $2^n$ different solutions. Balas' algorithm is, in effect, a set of rules by which an optimal solution can be obtained by examining only part of the branches of this tree of $2^n$ branches. The algorithm retains $z^*$, which is the minimum value of the objective function found previously for any basic feasible solution. In considering other solutions, the algorithm will abandon any branches that will lead to less attractive values of the objective function. Thus, if a solution has a value of the objective function $z_s^*$, then we need not consider any variables $j$ for inclusion in the next iteration for which $c_j \geq z^*_s - z^*_s$. In other words, the present value of $z^*$ is a ceiling for solution $u^s$.

Initially $z^* = \infty$.

If we identify those $x_j$'s which are equal to one in the current solution as a set $J_s$ which is contained in the original set $N$ of all $x_j$'s, then

$$y_i = b_i - \sum_{j \in J_s} a_{ij}$$
C^s is defined as the set of those variables which have been eliminated from consideration on this branch.

D_s is defined as those variables in (N-C^s) for which

\[ c_j \geq z^* - z_k^* \, . \]

E_s is the set of x_j such that for all y_i < 0 all a_{ij} \geq 0 .

We can now define a set of "improving vectors" (i.e., variables which are candidates to improve solution u^s).

\[ N_s = N - C^s \cup D_s \cup E_s \, . \]

At each solution level on the tree we compute for all j in

\[ N_s \]

\[ v_j = \begin{cases} 
(y_i - a_{ij}) & \text{where } y_i - a_{ij} \leq 0 \\
0 & \text{otherwise} 
\end{cases} \]

The maximum value of v_j is selected to indicate which variable is to enter the solution. The v_j selected is dropped from N_s (thus added to C^s). Whenever N_s becomes empty at any one level we must look back to an earlier level to see if an improvement can be made on z^*. Of course, if all N_s are empty and z^* = oo , there is no feasible solution.

The cycling back through the tree is done in an orderly fashion.

N_k (k < s) for previously computed solutions for which the set

J_k \subset J_s \, are reduced by defining
\[ C_k^s = J_s - J_k \text{ for any } j \text{ in } N_k \]

\[ D_k^s = \{ j \mid j \notin (N_k - C_k^s), c_j \geq z^* - z_k \} \]

and

\[ N_k^s = N_k - (C_k^s \cup D_k^s) \]

where \( N_k^s \) is the improving vector for iteration \( k \) following solution \( s \). Whenever the improving vectors for a solution \( u^s \) is void, the algorithm has encountered a "stop signal" which means that there is no feasible solution \( u^t \) such that \( J_s \subseteq J_t \) and \( z_t < z^* \).

Readers interested in a complete statement of the algorithm as well as some samples of the computations may refer to Balas' paper (3). The author has implemented a generalized computer program of Balas' algorithm.

The hypothesis of this thesis is that if a computer program for the solution of linear programs involving variables which take on only the values zero or one is available, then the task of scheduling graduate assistants to chemistry laboratories at Oregon State University can be formulated to use that computer program.

The graduate assistant scheduling problem will be formulated in accordance with Balas' algorithm in the next chapter.
CHAPTER V

FORMULATION OF THE SCHEDULING OF CHEMISTRY DEPARTMENT GRADUATE ASSISTANTS AS A LINEAR PROGRAM

The assignment of graduate assistants to teach laboratory sections in the Oregon State University Chemistry Department is an example of a zero-one integer linear programming problem. If the relationship of each graduate assistant to each lab section is treated as a single variable, then there is a dichotomous choice regarding each possible assignment. For ease of illustration we will use a double subscript for the individual items of \( x \) and \( c \), where \( x_{ij} \) will correspond to the relationship of student \( i \) to lab \( j \).

\[
x_{ij} = 1 \text{ if } i \text{ assigned to } j
\]

\[
x_{ij} = 0 \text{ if } i \text{ not assigned to } j
\]

\( c_{ij} \) will be a utility function assigned to \( x_{ij} \). Values that might be used for \( c_{ij} \) are:

1. if department prefers assignment
2. if student requests assignment
3. if no preference
4. if student requests "no assignment"
5. if department requests "no assignment"
6. 999 for "fixed charge" cost
7. 99999 \((M)\) if assignment must be avoided.
The "fixed charge" is a result of a requirement by the Chemistry Department that, for those students who have multiple assignments, the assignments should all be made in the same course (for instance, Ch 101). Thus, for each graduate assistant, if there are m different courses and \( \ell \) total lab sections, then \( 1 \leq j \leq \ell + m \). The manner by which these "fixed charges" are used to force the desirable assignment is illustrated in constraint type #3 below. For a complete discussion of "fixed charge" problems see Dantzig (7, p. 545-547), or Hadley (12, p. 252-253).

There are four types of constraints:

1. Student loads, where for each student

\[ \sum_{j} h_{j}x_{j} \leq s_{i} \]

where

- \( h_{j} \) = hours of work associated with lab \( j \)
- \( s_{i} \) = student's maximum load

2. Lab staffing, where for each lab section

\[ \sum_{i} x_{i} \geq w_{j} \]

where

- \( w_{j} \) = number of assistants needed by lab \( j \)

(Note that these constraints must be converted to \( \leq \) form.)

3. Fixed charge, where for each student there are \( m \) inequations, each involving the set of \( x_{j} \)'s included in
a single course. Thus for the kth course
\[ \sum x_j - 50x_{\ell+k} \leq 0 \quad \text{for all } j \text{ in } k \]

(4) Lab time conflicts, representing the sets of labs that overlap in meeting time. For each student there will be a set of constraints, each representing a conflict set \( p \)
\[ \sum x_j \leq 1 \quad \text{for all } j \text{ in } p \]

Graduate assistants vs. laboratory schedule conflicts are taken care of by assigning \( c_{ij} = M \) for all such conflicts.

The dimensions of the Chemistry Department problem are as follows:

<table>
<thead>
<tr>
<th></th>
<th>Graduate Assistants</th>
<th>Laboratory Sections</th>
<th>Courses</th>
<th>Total Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;M&quot; ( c_{ij} )'s</td>
<td></td>
<td></td>
<td></td>
<td>5490</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Constraints</th>
<th>Number</th>
<th>Non-Zero Elements Per Constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student Loads</td>
<td>45</td>
<td>117</td>
</tr>
<tr>
<td>Lab Staffing</td>
<td>117</td>
<td>45</td>
</tr>
<tr>
<td>Fixed Charge</td>
<td>225</td>
<td>24</td>
</tr>
<tr>
<td>Lab Conflicts</td>
<td>1080</td>
<td>10</td>
</tr>
<tr>
<td>Total</td>
<td>1467</td>
<td></td>
</tr>
</tbody>
</table>
From the table above we find that only one-third of one percent of the A-matrix is non-zero. Actually, there are at most only five non-zero elements in each of the 5490 columns, since each possible graduate assistant-laboratory section combination is affected by only one each of constraint types 1, 2 and 3 and at most by two of the lab conflict constraints.

While one-third of one percent represents a very sparse matrix, the non-zero elements still total 26,730 out of the full matrix of 8,053,830 elements. Because of the large number of elements to be dealt with in the A-matrix, the generalized computer program of Balas' algorithm must be modified to accommodate the problem. The sparseness of the matrix is encouraging since the time required to obtain a solution is directly related to the sparsity of the matrix (6, p. 3).

This chapter has outlined the formulation of the graduate assistant assignment to chemistry laboratory problems in accordance with Balas' algorithm. The sixth chapter will summarize the conclusions drawn from this study.
CHAPTER VI

CONCLUSIONS

Several test runs have been made on the graduate assistant scheduling problem using a modified version of the general program for performing Balas' algorithm. The principal changes in the program involve the storing of the non-zero A-matrix elements and their indices on magnetic tape. The general program, which is limited to processing 200 variables, retains the entire A-matrix in the memory of the computer.

The tests run on a Control Data Corporation 3300 computer indicate that problems of this type cannot be done economically on computers of this or comparable capability. The formulation of the problem, which includes generation of the A-matrix, the cost vector and the b-vector, requires at least 30 minutes of computer time. If the non-zero elements of the A-matrix are to be stored on magnetic tape or disk, it is desirable to have them stored in two different sequences, since some parts of the algorithm must deal with the A-matrix by columns while other sections deal with the same matrix by rows.

The running time per iteration of the problem with the A-matrix on magnetic tape is between two and ten minutes. This may
be reduced by using magnetic disk storage in place of the magnetic tape. However, since the number of iterations to obtain a single basic feasible solution must be at least 162 (117 laboratory assignments plus 45 fixed charge variables), the cost at one minute per iteration would seem to be prohibitive. To obtain an optimum solution may very well require several thousand iterations.

Glover and Zionts (9) have suggested some ways to reduce the number of solutions examined by Balas' method. Unfortunately, these suggestions involve changes which introduce division operations, greatly increasing the computer memory requirements.

Thus, the author concludes that while the instructor scheduling problem as exemplified by the Chemistry Department graduate assistant laboratory assignment problem can be formulated as a linear program, the economic solution of problems of this size requires a computer an order of magnitude faster than equipment currently available at Oregon State University.


