

In presenting the dissertation as a partial fulfillment of the requirements for an advanced degree from the Georgia Institute of Technology, I agree that the Library of the Institute shall make it available for inspection and circulation in accordance with its regulations governing materials of this type. I agree that permission to copy from, or to publish from, this dissertation may be granted by the professor under whose direction it was written, or, in his absence, by the Dean of the Graduate Division when such copying or publication is solely for scholarly purposes and does not involve potential financial gain. It is understood that any copying from, or publication of, this dissertation which involves potential financial gain will not be allowed without written permission.

A. C. P.
Dean of the Graduate Division

[Signature]

7/25/68

INTEGER PROGRAMMING FOR IMBEDDED UNIMODULAR CONSTRAINT MATRICES
WITH APPLICATION TO A COURSE-TIME SCHEDULING PROBLEM

A THESIS

Presented to

The Faculty of the Division of Graduate
Studies and Research

by

William Walter Swart

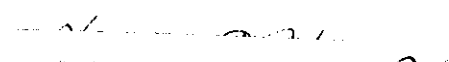
In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
in the School of Industrial and Systems Engineering

Georgia Institute of Technology

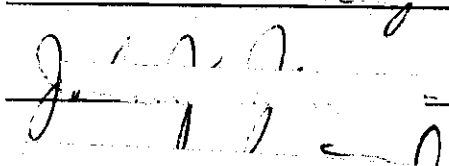
July, 1970

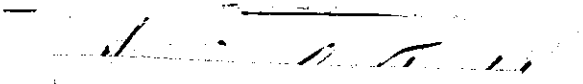
INTEGER PROGRAMMING FOR IMBEDDED UNIMODULAR CONSTRAINT MATRICES
WITH APPLICATION TO A COURSE-TIME SCHEDULING PROBLEM

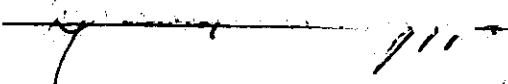
Approved:



Chairman







Date approved by Chairman: 8/25/70

ACKNOWLEDGMENTS

I wish to express my gratitude to Dr. V. E. Unger for his many contributions to the work presented in this thesis. His guidance and direction have been invaluable and are deeply appreciated.

Special thanks are also due to Dr. R. N. Lehrer, Director of the School of Industrial and Systems Engineering, for his support of this work and for the many ways he has helped me during my stay at the Georgia Institute of Technology.

To Dr. D. E. Fyffe I wish to express my thanks for the inspiration he provided, especially during my first three years at this institution. I also wish to thank him together with Dr. J. J. Jarvis, Dr. R. N. Baker, and Dr. W. R. Smythe, Jr., for their helpful suggestions while serving on my reading committee.

One of the more fulfilling aspects of this work has been the opportunity of working and making friends with the students who took part in the development of ISEMIS. I wish to thank Marion Guerin, Bob Heiks, Bob Bulfin, and the other members of CAG for their assistance, friendship, and cooperation.

I also wish to thank Mrs. Betty Sims for the great job she has done in typing this thesis, and my sister-in-law, Miss Carmen Smith, for the many hours she spent editing and proofreading this thesis.

Last, I wish to thank my parents for their support, wisdom, and counsel, which have molded me, and my loving wife for the finishing touches which are still being applied.

TABLE OF CONTENTS

	Page
ACKNOWLEDGMENTS.	ii
LIST OF ILLUSTRATIONS.	iv
LIST OF TABLES	vi
SUMMARY.	vii
Chapter	
I. INTRODUCTION.	1
Areas of Applicability	
Objectives and Purpose	
Introductory Research and Review of the Pertinent Literature	
II. ALGORITHM DEVELOPMENT	33
Basic Principles	
The Algorithm	
Summary of Algorithm	
III. APPLICATION-FORMULATION OF A COURSE SCHEDULING PROBLEM. . .	63
ISEMIS	
The Coursched Subsystem	
Mathematical Aspects	
Relevance of Partitioning Schemes	
IV. APPLICATION—COMPUTATIONAL ASPECTS OF THE COURSE SCHEDULING PROBLEM	100
Data Organization	
The Branch Help Table	
Computational Analysis and Experience	
Computational Synthesis	
V. CONCLUSIONS AND RECOMMENDATIONS	130
BIBLIOGRAPHY	143
VITA	145

LIST OF ILLUSTRATIONS

Figure	Page
1. Summary Results from Initial Steps of Algorithm	45
2. General Results of Algorithm at Some Intermediate Level of Computation	59
3. Summary Presentation of Algorithm	61
4. Conceptual Representation of ISEMIS	64
5. Coursched in Relation to ISEMIS	68
6. Constraint Coefficient Matrix: Course-Time Scheduling Problem.	73
7. Constraint Coefficient Matrix of Imbedded Problem: SIP Partitioning Scheme	79
8. Network Representation of Imbedded Problem: SIP Partitioning Scheme	80
9. Constraint Coefficient Matrix of Imbedded Problem: MIP Partitioning Scheme	87
10. Network Representation of Imbedded Problem: MIP Partitioning Scheme	88
11. Network Representation of Imbedded Problem: LIP Partitioning Scheme	94
12. Computations Required at Level Zero of BHT.	109
13. Application of Branching Rule and Bounding of Subsets--Part I	111
14. Application of Branching Rule and Bounding of Subsets--Part II.	113
15. Decision Process for Choice of Subset for Further Partitioning	114
16. Backtrack Logic	116

Figure	Page
17. Computational Synthesis Scheme: LIP-MIP.	124
18. General Synthesis Scheme.	128
19. Conceptual Structure of University-Wide Course Scheduling Problem	136

LIST OF TABLES

Table	Page
1. Dimensionality of Course Scheduling Problem for Different Partitioning Schemes	98
2. Data Input Sequence Description	102
3. Relation of BHT Entries to Computational Sequences.	115
4. Summary Presentation of Problem One with Solution Times . .	118
5. Summary Presentation of Problem Two with Solution Times . .	119
6. Summary Presentation of Problem Three with Solution Times	120
7. SIP Partitioning Scheme: Detailed Solution Patterns and Times	121
8. MIP Partitioning Scheme: Detailed Solution Patterns and Times	122
9. LIP Partitioning Scheme: Detailed Solution Patterns and Times	123
10. Summary Results: Synthesis of LIP and MIP Partitioning Schemes.	125

SUMMARY

This thesis develops a solution methodology for the problem:

minimize CX

subject to:

$$A_1 X = b_1$$

$$A_2 X \leq b_2$$

X = Vector with integer components

where:

C - Row vector with n integer components.

X - Column vector with n components.

A_1 - m_1 by n matrix possessing the unimodular property.

A_2 - m_2 by n matrix with integer components.

b_1 - Column vector with m_1 integer components.

b_2 - Column vector with m_2 integer components.

The solution methodology is based on the principles of implicit enumeration and takes advantage of the structure of the matrix A_1 for computing lower bounds.

Models with the above structure arise in many areas of application. Examples of some of these are multicommodity network flows, multi-period decision making, and scheduling.

The algorithm developed is general in nature, but requires that the imbedded problem be converted to a zero-one integer programming problem. A method is suggested in the thesis to make the required conversion, but the conversion increases the number of variables in the imbedded problem as a function of the values of the components of b_1 . The algorithm is particularly suited for situations in which the components of b_1 are small.

Computational results for the algorithm are obtained for the class of course-time scheduling problems formulated in the thesis. The results obtained serve to illustrate the computational aspects of the algorithm.

Approaches to extend the work of this thesis are outlined in some detail. One extension suggested is the application of the methodology to other classes of problems. Another is the extension of the course scheduling model to incorporate university-wide course scheduling. For the latter, a stage-wise solution procedure is outlined based on the methodology developed earlier in the thesis.

CHAPTER I

INTRODUCTION

This thesis deals with the development of an algorithm for integer programming problems which have imbedded in their constraint set a subset of constraints whose coefficient matrix possesses the unimodular property. Although the algorithm is applicable to a large number of problems, the particular application utilized in this thesis is to a course-time scheduling problem.

The results of the application will demonstrate the computational aspect of the algorithm and the computational efficiencies that can be obtained.

This chapter is subdivided in three sections. The first section will present some areas in which problems of the type for which an algorithm will be developed occur. The second section will state the objectives and purpose of the research, and the last section will present a review of the pertinent literature together with some results of the introductory research.

Areas of Applicability

Multicommodity Network Flows

The general multicommodity network flow problem may be stated as:

$$(1) \text{ minimize } \sum_M \sum_N \sum_P c_{ijk} x_{ijk}$$

subject to:

$$(2) \quad \sum_N x_{ijk} = a_{ik} \quad \forall i \in M, j \in N, k \in P$$

$$(3) \quad \sum_M x_{ijk} = b_{jk} \quad \forall i \in M, j \in N, k \in P$$

$$(4) \quad 0 \leq x_{ijk} \leq d_{ijk} \quad \forall i \in M, j \in N, k \in P$$

$$(5) \quad \sum_P x_{ijk} \leq D_{ij} \quad \forall i \in M, j \in N, k \in P$$

$$(6) \quad x_{ijk} = \text{binary} \quad \forall i, j, k$$

where:

$M = \{1, \dots, i, \dots, m\}$: origins

$N = \{1, \dots, j, \dots, n\}$: destinations

$P = \{1, \dots, k, \dots, p\}$: commodities

a_{ik} = amount of k th commodity available at origin i .

b_{jk} = amount of k th commodity required at destination j .

c_{ijk} = cost of shipping one unit of the k th commodity from origin i to destination j .

d_{ijk} = maximum amount of the k th commodity that can be shipped from origin i to destination j .

D_{ij} = maximum amount of all commodities that can be shipped from origin i to destination j .

x_{ijk} = number of units of k th commodity shipped from origin i to destination j .

For this formulation of the multicommodity flow problem, constraint sets (2), (3), and (4) together with the objective function constitute a set of P independent capacitated transportation problems for which efficient methods of solution are available. For example, see [8], [12], or [14]. Constraints (5) create dependencies between various variables in the otherwise independent capacitated transportation problems, and consequently prevents the use of the efficient solution techniques available.

Bozoki [8] discusses several applications for which the multi-commodity network flow formulation is an appropriate model.

Multi-Copy Generalized Networks and Multi-Page Programs

A multi-page formulation may be thought of as derived from the coupling of many component programs. For example, the individual component programs may associate with one planning period, and the combined activities for all planning periods may be further constrained by overall considerations such as limited availability of material. In some instances a multi-copy designation is used because each component program may have the same form albeit with different cost and constraint coefficients. With respect to the above, Charnes and Cooper [10], and Charnes and Lemke [11] have done work in linear programming relating to models which have constraint systems which are not arbitrary in character, but which consist of portions which individually belong to a relatively small number of model types. For some of these, efficiencies and capabilities greatly surpassing the computational efficiencies and capabilities associated with arbitrary models, e.g., network problems, generalized network problems, etc., can be achieved.

The work done in this area by the above authors was in improving the efficiencies over those obtainable using standard linear programming methodology. For many of the situations discussed, integer programming models would be more appropriate.

The general formulation of the model in question is:

$$\text{minimize} \quad \sum_{p=1}^P c_p^T x_p$$

subject to:

$$A_p x_p = b_p \quad p = 1, P$$

$$\sum_{p=1}^P B_p x_p + Iy = b_o$$

$$x_p, y \geq 0 \quad \forall p$$

where:

- A_p - Constraint coefficient matrix associated with the pth page program, generally with some "favoured" structure.
- x_p - Decision vector associated with the pth page program.
- b_p - Right-hand side vector associated with pth page program.
- B_p - Coefficient matrix of the decision variables of the pth page program appearing in the overall constraints.
- y - Vector of slack variables associated with overall constraints.
- b_o - Right-hand side vector of overall constraint.

c_p - Vector of cost coefficients associated with the decision variables of the pth page program.

Situations in which models of the above type are adequate have been described in [11]. Some of these are:

Models to predict traffic flows in an arterial network.
 Models to optimize chemical processing.
 Models to analyze behavior of loaded structures.
 Models to optimize finances and production over time.

Course-Time Scheduling Problems

As part of an integrated management information and decision system for the School of Industrial and Systems Engineering of the Georgia Institute of Technology, the author has developed the following model to schedule courses to time slots:

$$\text{minimize} \quad \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij}$$

subject to:

$$\sum_{i=1}^m x_{ij} = 1 \quad j=1, n$$

$$\sum_{j \in D_k} x_{ij} \leq M_i^k \quad i=1, m; k=1, m_1$$

$$\sum_{j \in W_k} w_{ij} \leq 1 \quad i=1, m; k=1, m_2$$

$$\sum_{j \in L_k} x_{ij} \leq 1 \quad i=1, m; k=1, m_3$$

$$x_{ij} = \text{binary} \quad i=1, m; j=1, n$$

where:

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

c_{ij} - Cost measure of teaching course j at time i .

D_k - Set of courses with forecasted enrollment $\leq U_k$.

M_i^k - Number of classrooms with capacity $\geq U_k$.

W_k - Set of courses taught by professor k .

L_k - Set of courses which, if taught concurrently, would create conflicts for students.

The following chapter will present an algorithm which can be utilized to solve problems of the type presented in the preceding section. The computational results that can be achieved with this algorithm will be illustrated by using the course-time scheduling problem presented above.

The previous sections have presented several situations for which the appropriate mathematical model consists of a set of constraints which has imbedded in it a subset of constraints whose constraint coefficient matrix possesses the unimodular property. If it were possible to only consider the imbedded constraint set, then a solution to the model could be obtained by very efficient means.

Objectives and Purpose

The objectives of this thesis are:

(1) To develop a general solution strategy for integer programming problems which have imbedded in their constraint set a subproblem whose constraint matrix is unimodular.

(2) To apply this solution strategy to the course-time

scheduling problem and to illustrate the computational aspects of the solution strategy using the scheduling problem as a basis.

The purpose of the research is to achieve the above objectives by taking advantage of the unimodular property of the imbedded problem.

Introductory Research and Review of Pertinent Literature

Many methods have been suggested in the literature to solve linear programming problems which have imbedded in their constraint sets subproblems which, if isolated, could be solved very efficiently. A logical approach toward developing solution strategies for integer programming problems which exhibit these same characteristics is to attempt to extend the procedures developed for linear programming to include integer programming problems. The next section will be devoted to the presentation of some of the methodologies available for linear programs, and some discussion of the difficulties that arise when they are extended to include integer programs.

From a theoretical standpoint, one of the more elegant approaches developed for handling the type of problem being discussed is that of decomposition. Dantzig and Wolfe [13] present it in the context of solving the problem:

$$\text{minimize} \quad \sum_{i=1}^k C_i^T x_i$$

subject to:

$$\sum_{i=1}^k A_i x_i = b_0$$

$$B_i x_i = b_i \quad i=1,k$$

$$x_i \geq 0 \quad i=1,k$$

where A_i is an m_0 by n_i matrix, B_i an m_i by n_i matrix, C_i , x_i , and b_i are n_i component vectors, and b_0 is an n_0 component vector.

The basic concept of solving the above problem by Dantzig-Wolfe decomposition is that any solution to the complete problem must satisfy the constraint sets $B_i x_i = b_i$, and any point satisfying these can be expressed as a convex combination of the extreme points of that constraint set. Therefore, the problem may be rewritten as:

$$\text{minimize} \quad \sum_{i=1}^k C_i^T \sum_{j=1}^{E_i} \lambda_{ij} x_{ij}^*$$

subject to:

$$\sum_i A_i \sum_{j=1}^{E_i} \lambda_{ij} x_{ij}^* = b_0$$

$$\sum_{j=1}^{E_i} \lambda_{ij} = 1 \quad i=1,k$$

$$\lambda_{ij} \geq 0 \quad \forall i \text{ and } \forall j$$

where:

λ_{ij} - Convex combination constants for i th subproblem.

E_i - Number of extreme points in i th subproblem.

Although it appears that before the above problem can be solved, all the extreme points must be known, in actuality these can be generated as required. The mechanism developed to achieve this is derived from duality considerations. Namely, at any intermediate step, the simplex multipliers can be classified according to whether they are associated with the constraint set $\sum_i A_i \sum_{j=1}^{E_i} \lambda_{ij} x_{ij}^*$, or with the convexity constraints $\sum_{j=1}^{E_i} \lambda_{ij} = 1 \quad \forall i$.

To determine whether optimality has been achieved, the quantity

$$C_i^T x_{ij}^* - [\pi_{m_0} \pi_k] \begin{pmatrix} A_i x_{ij}^* \\ e_j \end{pmatrix} \quad \forall j \notin \text{Basis}$$

must be calculated, where π_{m_0} are the simplex multipliers associated with the first m_0 rows, and the π_k the multipliers associated with the last k rows. e_j indicates the entry due to the convexity constraints.

The above quantity may be rewritten as:

$$C_i^T x_{ij}^* - [\pi_{m_0} A_i x_{ij}^* + \pi_k e_j] = [C_i^T - \pi_{m_0} A_i] x_{ij}^* - \pi_j$$

if there exists some x_{ij}^* for which this quantity is negative, then optimality has not been achieved, and λ_{ij} should enter the basis.

Since all the x_{ij}^* are not known, the above determination can be made by solving:

$$\text{minimize} \quad [C_i^T - \pi_{m_0} A_i] x_i$$

subject to:

$$B_i x_i = b_i$$

$$x_i \geq 0$$

for each i .

When the structure of the constraint set B_i is such that efficient procedures exist to solve the above problem, the use of decomposition to solve large scale problems may indeed be very fruitful although, in general, it has been found that from a computational standpoint it is somewhat inefficient.

As the first step in the introductory research, the possibilities of combining the Dantzig-Wolfe decomposition approach with one of the existing cutting plane techniques was investigated. The particular problem which was considered was the course-time scheduling problem presented earlier. The formulation of this problem could be presented as:

$$\text{minimize} \quad \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij}$$

subject to:

$$(1) \quad \sum_{j \in L_k} x_{ij} \leq 1 \quad k=1, m_3; \quad i=1, m$$

$$\begin{aligned}
 (2) \quad & \sum_{j \in D_k} x_{ij} \leq M_i^k && k=1, m_1; i=1, m \\
 (3) \quad & \left. \sum_{i=1}^m x_{ij} = 1 \right\} && \\
 (4) \quad & \left. \sum_{j \in W_k} x_{ij} \leq 1 \right\} && j \in W_k, k=1, m_2; i=1, m \\
 (5) \quad & x_{ij} = \text{binary} && \forall i, j
 \end{aligned}$$

where the constraints defined by (3), (4), and (5) form the subproblems, and the others are overall constraints.

The above formulation has the interesting property that any solution to the overall problem is composed of one extreme point from each subproblem. As such, the optimal solution would be one in which each λ_{ij} in the decomposition formulation assumes a binary value. Since the convexity constraints insure that no value of λ_{ij} can be greater than one, it is sufficient to specify that the master problem must be solved as an integer programming problem. The basic reason for considering the combination of decomposition with a cutting plane approach stems from the fact that decomposition relies heavily on duality theory in that the master problem supplies the simplex multipliers required for the objective functions of the subproblems. If the master problem were to be solved by other methods, the duality relationships would be lost, while if cutting planes are generated, all duality relationships obtained from the master problem still hold when an integer solution has been obtained.

Gomory's [16] cutting plane approach to solving integer programming problems has received considerable attention in the literature. This approach is dual in nature since it starts with an optimal solution to the linear programming problem, and then sequentially generates new constraints or cutting planes, which render the previous solution infeasible. These cutting planes have the following properties: (1) Generated constraints never exclude from the new feasible region an integer feasible point of the original problem. (2) In a finite number of steps enough constraints will be generated so that the optimal solution to the modified linear program has integer components. (3) The generated constraints pass through at least one integer point.

The cutting planes described by Gomory are obtained as a linear combination of the non-basic variables. Since, in general, the type problems solved by decomposition are large and solved through the use of the revised simplex algorithm, the necessary information to generate cut equations is not readily available.

In conjunction with investigating decomposition and integer programming, it was hypothesized that perhaps it would be fruitful to determine whether Gomory cuts could be generated implicitly. Basically, the question was centered on the issue that, since columns are generated for the master problem by minimizing the dual evaluators, the simplex multiplier associated with a cut equation could be expressed in terms of the objective function associated with a particular subproblem. If this were possible, then it would not be necessary to have an explicit representation of the columns associated with the non-basic variables.

Since the Gomory cutting plane algorithm relies on the dual simplex method, the above scheme would not be directly applicable. One manner in which to avoid using the dual simplex method is to resort to a primal cutting plane approach. Young [20] has proposed such an approach. In order to briefly explain the approach, it will be assumed that the initial problem is in canonical form, i.e.:

$$I_{m+1}^- x_B + \bar{A} x_N = \bar{A}_0$$

where x_B is a column vector of x_i ($i=0, \bar{m}$); x_N is a column vector of x_j ($j=\bar{m}+1, \bar{n}$); I_{m+1}^- is an identity matrix of dimension $\bar{m}+1$; \bar{A}_0 is a column vector of \bar{a}_{i0} ($i=0, \bar{m}$); and \bar{A} is a $\bar{m}+1$ by $\bar{n}-\bar{m}$ matrix composed of the constraint equation coefficients after these have been put in canonical form. The bars above the various terms serve to indicate that these are the generic terms of the original system.

At some intermediate stage of computation, the existing tableau will be denoted by:

$$I_{m+1}^- x_B + AU_n = A_0$$

The above representation differs from the original in that it includes new variables and equations which were added to the original system. The added equations are cut equations similar to Gomory cuts that were added in order to preserve the integrality of the system, and the variables are the slacks associated with each cut equation.

The cutting plane approach developed by Young is primal in nature, and it requires that $\bar{A}_0 \geq 0$ and that \bar{A} and \bar{A}_0 be integer. The cutting plane approach will perpetuate these properties in A and A_0 .

At some intermediate stage of computation, the following cut equation is generated and added to the system:

$$S + \sum_{j \in N} \left\langle \frac{a_{vj}}{\lambda} \right\rangle u_j = \left\langle \frac{a_{vo}}{\lambda} \right\rangle$$

where a_{vj} is the entry in row v and column j of A , a_{vo} the entry in row v of A_0 , and $\lambda = a_{vj}$. This cut is derived as follows:

1. Calculate pivot ratio:

$$\theta_J = \min_{a_{ij} > 0} \left(\frac{a_{io}}{a_{iJ}} \right)$$

where J is index of some column A_J for which $a_{oJ} < 0$.

2. Select a source row for a Gomory cut some row v for which:

$$0 \leq \left\langle \frac{a_{vo}}{a_{vJ}} \right\rangle \leq \theta_J$$

The variable S in the cut becomes a basic variable with value $\left\langle \frac{a_{vo}}{a_{vJ}} \right\rangle$ in the current basic solution.

In the above $Y = \langle x \rangle \Rightarrow Y$ is largest integer $\leq x$.

Although the previous was only a very general outline of some basic concepts used by Young, it is enough to illustrate the situation encountered when this approach is used in connection with decomposition.

Recall that at some intermediate stage of computation of the decomposition algorithm, the variable to enter the basis is determined from the solution of the problem:

$$\text{minimize} \quad [C^T - \pi_m A_i] x_i$$

subject to:

$$B_i x_i = b_i$$

From the solution of the above, the coefficient vector of the variable to enter the basis can be obtained in terms of the present basis of the master problem. Denote this vector by A_J , i.e.:

$$A_J = B^{-1} \begin{bmatrix} A_i x_{ij}^* \\ \delta_x \end{bmatrix}$$

where B^{-1} is the inverse of the current basis of the master problem.

A_i is the matrix of coefficients associated with the vector x_i in the original problem, x_{ij}^* is the solution to the subproblem, and δ_x is a vector with zeros everywhere except in the i th position.

Following Young's procedure, a cut equation of the form:

$$S + \sum_{j \in N} \left\langle \frac{a_{vj}}{\lambda} \right\rangle u_j = \left\langle \frac{a_{v0}}{\lambda} \right\rangle; \text{ with } \lambda = a_{vJ}$$

is appended to the master problem. Since only a_{vJ} is known, it would be hard to immediately find all the terms of the cut equation. For the immediate purpose it is possible to append the slack S to the basis of the master problem, and augment A_J by the term corresponding to it from the cut equation. This would yield as the new pivot column:

$$A_J^* = \begin{bmatrix} A_J \\ 1 \end{bmatrix}$$

and as the new basis:

$$B^* = \begin{bmatrix} B & 0 \\ 0 & 1 \end{bmatrix}$$

and a change of basis may now be made with A_J^* as the pivot column, and the newly appended equation as the pivot row.

To determine optimality of the new solution, the dual evaluators must be computed for all non-basic variables. After a cut is adjoined to the system, the coefficient vector of each non-basic variable will have the form:

$$B^{*-1} \begin{bmatrix} A_i & x_{ij}^* \\ \delta_x \\ \sigma_x \end{bmatrix}$$

where σ_x are the coefficients added to a column of the master problem by the cut equations so far generated, and are a function of the extreme point being considered in the particular column. Partitioning the vector of simplex multipliers as follows:

$$[\pi_m \ \pi_k \ \pi_c]$$

where π_c are the multipliers for the cut equations so far adjoined to the master problem. The evaluators now become:

$$[C_i^T - \pi_m A_i] x_{ij}^* - \pi_c \sigma_x - \pi_j$$

in which each term of σ_x would have the largest integer operator as part of it, and the argument of this operator being a function of x_{ij}^* .

Therefore, before an additional iteration can be performed, a problem of the form:

$$\text{minimize} \quad [C_i^T - \pi_m A_i] x_i - \pi_c \sigma_x$$

subject to:

$$B_i x_i = b_i$$

must be solved. The above problem is not trivial to solve since the term σ_x is of the form:

$$\sigma_x = \langle [a_1 \ a_2 \ \dots \ a_n][x_{1i} \ x_{ni}]^T \rangle$$

in which the a_i terms correspond to elements of the master problem row utilized in generating a cut, and the x_{ji} are the decision variables of the problem. Here again, $y = \langle x \rangle \Rightarrow y$ largest integer $\leq x$.

As can be noted, the objective function of the problem that must be solved is non-linear and consequently, the solution of the problem is non-trivial. Since many such problems would have to be solved during a cutting plane-decomposition approach, further investigation was discontinued.

Charnes and Cooper [10] have developed a procedure to solve problems of the form:

$$\text{maximize} \quad C^T \lambda$$

subject to:

$$P\lambda = P_0$$

$$R\lambda = b$$

$$\lambda \geq 0$$

where P and R are matrices and C , λ , P_0 , and b are vectors of appropriate order. It is assumed that the constraint set $P\lambda = P_0$ has some favored structure in the sense that specialized and efficient techniques exist for their solution.

The procedure, denoted as the double reverse method, consists of writing a basic solution to the above problem as:

$$\begin{bmatrix} B_F & B_G \\ R_F & R_G \end{bmatrix} \begin{bmatrix} \lambda_F \\ \lambda_G \end{bmatrix} = \begin{bmatrix} P_0 \\ b \end{bmatrix}$$

where B_F is a basis for the favored problem, and R_F is the row-wise continuation of the columns associated with B_F ; finally, B_G and R_G are the completion of the basis for the entire problem.

The above expression may be rewritten in the form:

$$1) \quad B_F \lambda_F + B_G \lambda_G = P_0$$

$$2) \quad R_F \lambda_F + R_G \lambda_G = b$$

Since B_F is a basis for the favored problem, there exists a unique matrix, x_G , such that:

$$B_F x_G = B_G$$

substituting the above into 1) yields:

$$B_F [\lambda_F + x_G \lambda_G] = P_0$$

or,

$$B_F \hat{\lambda} = P_0$$

where:

$$\hat{\lambda} = \lambda_F + x_G \lambda_G$$

To find the values of the corresponding (basic) variables, it can be noted that:

$$\lambda_F = \hat{\lambda} - x_G \lambda_G$$

and, from 2)

$$R_F[\hat{\lambda} - x_G \lambda_G] + R_G \lambda_G = b$$

$$\Rightarrow [R_G + R_F x_G] \lambda_G = R_F \hat{\lambda} - b$$

or,

$$\lambda_G = [R_G + R_F x_G]^{-1} (R_F \hat{\lambda} - b)$$

Associated with this basis is a vector of dual evaluators, $\omega^T = [\omega_B^T, \omega_R^T]$

which uniquely satisfies:

$$[\omega_B^T, \omega_R^T] \begin{bmatrix} B_F & B_G \\ R_F & R_G \end{bmatrix} = [C_F^T, C_G^T]$$

where $[C_F^T, C_G^T]$ are, respectively, favored and non-favored entries in C^T .

That is, C_F^T is associated with λ_F and C_G^T is associated with λ_G in the original problem statement.

Rewriting the above expression as:

$$\omega_{B_F}^T + \omega_{R_F}^T = C_F^T$$

$$\omega_{B_G}^T + \omega_{R_G}^T = C_G^T$$

makes a similar procedure used to find the values of the basic variables applicable to finding the values of the dual evaluators. To show this, let:

$$\hat{\omega}_{B_F}^T = C_F^T$$

then:

$$\omega_R^T [R_F x_G - R_G] = \hat{\omega}_{B_G}^T - C_G^T$$

and:

$$\omega_B^T = \hat{\omega} - \omega_{R_Y}^T$$

where:

$$Y_{R_F} B_F = R_F$$

From the knowledge of the vector ω^T it is possible to determine what vector should enter the basis. Denoting the vector by:

$$[P_K^T \quad R_K^T]^T$$

then, in order to determine the vector to leave the basis, it is necessary to represent the above vector in terms of the present basis. This may be accomplished by solving:

$$\begin{bmatrix} B_F & B_G \\ R_F & R_G \end{bmatrix} \begin{bmatrix} \mu_F \\ \mu_G \end{bmatrix} = \begin{bmatrix} P_K \\ R_K \end{bmatrix}$$

where $[\mu_F^T \ \mu_G^T]^T$ is the representation of $[P_K^T \ R_K^T]^T$ in terms of the current basis.

The above may be solved by the previously defined steps of the double reverse method. Once μ_F and μ_G have been obtained, then the vector to leave the basis is found in the same manner as with the revised simplex method.

With the information of which vector is to enter the basis and which is to leave it, a new basis has been specified and the same process is repeated until termination.

Since the double reverse method's primary advantage is to utilize the imbedded problem and its properties to duplicate the function of other linear programming approaches, its extension to allow for integrality would entail the same difficulties as with linear programming and, as a consequence, this approach was not taken to develop integer programming schemes. In relation to the above, Bakes [2] presents an approach similar to the double reverse method presented in the previous paragraphs.

Another approach developed independently by the author but found

later to be similar to an approach taken by Bozoki [8] to solve the multi-commodity network flow problem was considered. In order to present the basic ideas of this approach, consider the problem:

minimize Cx

subject to:

$$(1) \quad A_1 x = b_1$$

$$(2) \quad A_2 x \leq b_2$$

where:

x - Vector of binary components.

C - Row vector of cost coefficients.

A_1 - Constraint coefficient possessing unimodular property.

A_2 - Coefficient matrix of additional constraints, $A_2 \geq 0$.

b_1 - Right-hand side vector of constraints whose coefficient matrix possesses the unimodular property.

b_2 - Right-hand side vector of additional constraints.

Also, it is assumed that the matrix and vector dimensions are compatible for the problem presented.

For the preliminary investigations, it was assumed that the constraint set defined by (1) formed a transportation problem. The basic idea motivating the approach was that of the dual simplex method. Namely, obtain the solution to the problem:

minimize Cx

subject to:

$$A_1 x = b_1$$

$$x \geq 0$$

denoting the optimal solution to the above problem by x^* , then if $A_2 x^* \leq b_2$ the optimal solution has been obtained. Otherwise x^* comprises a solution which is optimal to the imbedded problem, but infeasible with respect to the additional constraints. Consequently, if a scheme could be devised which reduced a measure of the infeasibility with respect to the additional constraints at minimum cost, while at the same time the constraints of the imbedded problem remained satisfied, then the optimal solution would be obtained. Expressing the constraints defined by (2) as:

$$\sum_{vj} [A_2]_{ij} x_{ij} \leq b_{2i} \quad \forall i$$

then a measure of infeasibility for each constraint may be defined as:

$$S_i = \begin{cases} -b_{2i} + \sum_{vj} [A_2]_{ij} x_{ij}^* & \text{if } b_{2i} - \sum_{vj} [A_2]_{ij} x_{ij}^* < 0 \\ 0 & \text{otherwise.} \end{cases}$$

and the measure of aggregate infeasibility may be defined as:

$$I = \sum_{\forall i} S_i$$

Assuming that the imbedded problem is, or has been, transformed to a zero-one integer program (see Chapter V for a method of transforming certain classes of general integer programs to zero-one integer programs), then the problem of reducing the aggregate infeasibility, I , can be resolved by finding all the x_{ij}^* which have a value of one, and from these select one such that if it were forced to a value of zero, the new basic solution obtained from the imbedded problem would reduce the measure of aggregate infeasibility. If there exists more than one such basis change, then the one selected would be that which increased the objective function the least. The above process would then be repeated until $I = 0$.

As can be noted from the above description, this approach utilizes the additional constraints to determine which variable should leave the basis obtained from solving the imbedded problem, but actually utilizes the imbedded problem to perform the basis change. In this manner it is always insured that integrality is maintained. Also, each basis change is performed at minimum incremental cost. Unfortunately, the dependencies that exist between basic and non-basic variables create a situation in which, even though it is possible to make a basis change at minimum incremental cost at each iteration, at the termination of the process the sum of the minimum cost basis changes performed does not, in general, yield a solution which has reduced the measure of aggregate infeasibility to zero at minimum total incremental cost. Due to this difficulty, the above described approach was abandoned.

A logical extension of the above approach, which is iterative in nature, would be the investigation of the problem that is obtained by attempting to reduce infeasibility at minimum incremental cost. Again consider the problem:

minimize Cx

subject to:

$$A_1 x = b_1$$

$$A_2 x \leq b_2$$

x = Vector with binary components.

Consider that a basic solution to the imbedded problem is available, and denote x_B as the set of basic variables and x_D as the set of non-basic variables. Further, partition the problem as follows:

minimize $C_B x_B + C_D x_D$

subject to:

$$(1) \quad A_{1B} x_B + A_{1D} x_D = b_1$$

$$(2) \quad A_{2B} x_B + A_{2D} x_D \leq b_2$$

x = Vector with binary components.

where

A_{1B} - Coefficient matrix of basic variables of imbedded problem constraints.

A_{1D} - Coefficient matrix of non-basic variables of imbedded problem constraints.

A_{2B} - Coefficient matrix of basic variables of additional constraints.

A_{2D} - Coefficient matrix of non-basic variables of additional constraints

From (1), the basic variables may be represented in terms of the non-basic variables as:

$$x_B = A_{1B}^{-1}b_1 - A_{1B}^{-1}A_{1D}x_D$$

substituting the above in (2) yields:

$$[A_{2D} - A_{2B}A_{1B}^{-1}A_{1D}]x_D = b_2 - A_{2B}A_{1B}^{-1}b_1$$

and the objective function in terms of the non-basic variables becomes:

$$C_B A_{1B}^{-1}b_1 - [C_B A_{1B}^{-1}A_{1D} - C_D]x_D$$

with the above, the original problem may be restated as:

$$\text{maximize} \quad [C_B A_{1B}^{-1}A_{1D} - C_D]x_D$$

subject to:

$$(1) \quad [A_{2D} - A_{2B} A_{1B}^{-1} A_{1D}] x_D = b_2 - A_{2B} A_{1B}^{-1} b_1$$

$$(2) \quad A_{1B}^{-1} A_{1D} x_D \leq A_{1B}^{-1} b_1$$

$$(3) \quad A_{1B}^{-1} A_{1D} x_D \geq A_{1B}^{-1} b_1 - \bar{1}$$

x_D = vector with binary components

where $\bar{1}$ = vector with all components equal to one.

The constraints defined by (2) and (3) serve to insure that the basic variables remain binary.

After studying the above problem structure, it was concluded that there was no evidence that anything could be gained by concentrating on solution methods for such problems as opposed to the original problem.

As previously pointed out, the course scheduling problem can be structured in block diagonal form. For the purpose of the next paragraphs, consider a general representation of the problem in block diagonal form to be:

$$\text{minimize} \quad C_1 x_1 + C_2 x_2$$

subject to:

$$A_1 x_1 + A_2 x_2 \leq b_0$$

$$B_1 x_1 \leq b_1$$

$$B_2 x_2 \leq b_2$$

where:

B_1 - Coefficient matrix of subproblem one.

B_2 - Coefficient matrix of subproblem two.

x_1 - Vector of binary components associated with subproblem one.

x_2 - Vector of binary components associated with subproblem two.

A_1 - Coefficient matrix associated with the decision vector x_1 .

A_2 - Coefficient matrix associated with the decision vector x_2 .

C_1 - Row vector of cost coefficients associated with the decision vector x_1 .

C_2 - Row vector of cost coefficients associated with the decision vector x_2 .

b_i - Right-hand side vector of the i th constraint set.

If the restriction that the vectors x_i must consist of binary components is abandoned for the time being, then the dual of the above problem may be written as:

$$\text{minimize} \quad \mu_0 b_0 + \mu_1 b_1 + \mu_2 b_2$$

subject to:

$$\mu_0 A_1 + \mu_1 B_1 \geq C_1$$

$$\mu_0 A_2 + \mu_2 B_2 \geq C_2$$

$$\mu_i \geq 0$$

where μ_0 , μ_1 , and μ_2 are the vectors of dual variables. The above problem structure is ideally suited for solution utilizing a method developed by Benders [7], which is intended to solve mixed integer programming problems in which the variables represented by the vector μ_0 are restricted to being integer, while the remaining variables are continuous. The basic procedure consists of taking advantage of the separability property of the above formulation by noting that, for a fixed set of values for the components of the vector μ_0 , the problem would be:

$$\text{maximize} \quad \mu_0 b_0 + \mu_1 b_1 + \mu_2 b_2$$

subject to:

$$\mu_1 B_1 \geq C_1 - \mu_0 A_1$$

$$\mu_2 B_2 \geq C_2 - \mu_0 A_2$$

$$\mu_i \geq 0$$

or, equivalently:

$$\begin{aligned} & \mu_0 b_0 + \max_{\mu_1} \{ \mu_1 b_1 / \mu_1 B_1 \geq C_1 - \mu_0 A_1 \} \\ & + \max_{\mu_2} \{ \mu_2 b_2 / \mu_2 B_2 \geq C_2 - \mu_0 A_2 \} \end{aligned}$$

Taking the dual of the maximization problems yields:

$$\begin{aligned} & \mu_0 b_0 + \min_{x_1} \{ [C_1 - \mu_0 A_1] x_1 / B_1 x_1 \leq b_1 \} \\ & + \min_{x_2} \{ [C_2 - \mu_0 A_2] x_2 / B_2 x_2 \leq b_2 \} \end{aligned}$$

If the concept of having the above problem formulated for a fixed decision vector μ_0 is changed to having a vector of decision variables, then the original problem may be restated as:

$$\begin{aligned} & \text{maximize}_{\mu_0} \{ \mu_0 b_0 + \min_{x_1} \{ [C_1 - \mu_0 A_1] x_1 / B_1 x_1 \leq b_1 \} \\ & + \min_{x_2} \{ [C_2 - \mu_0 A_2] x_2 / B_2 x_2 \leq b_2 \} \}. \end{aligned}$$

Benders has suggested a solution procedure for the above type problem which utilizes the principles of implicit enumeration. It is of interest to note the similarity that exists between this approach and the decomposition approach.

Knowing that problems such as the above can be solved and, furthermore, that since the solutions of the subproblems would be integer, the overall solution would also be integer, the difficulty that presents

itself lies in what the relationship between solving a primal problem which is required to have integer solutions, and the solutions obtained by solving the dual of the original problem, neglecting the integrality conditions, as an integer program. Even though Balas [4-6] has done considerable work in the area of duality in integer programming, the results available do not help in resolving the difficulty.

CHAPTER II

ALGORITHM DEVELOPMENT

This chapter is divided into three sections. Section one will describe the general principles employed in developing a solution algorithm. The development of the section is based on a paper by Balas [2]. Section two will develop the general algorithm in detail, and section three will present the algorithm in summary form.

Basic Principles

Consider a finite set:

$$S = \{s_1, s_2, \dots, s_n\}$$

on which a function:

$$f: S \rightarrow R$$

is defined associating a real number $f(s_i) \in R$ with each $s_i \in S$. The problem is to find an optimal element $s_k \in S$ defined by:

$$f(s_k) = \min\{f(s_i) / s_i \in S\}$$

The basic principles embodied in the algorithm to be developed in later

sections of this chapter involve the definition of a finite superset S_1 of S :

$$S_1 = \{t_1 \ t_2 \ \cdots \ t_r\} \supset S$$

together with an extension g of f :

$$g: S_1 \rightarrow \mathbb{R}$$

associating a real number $g(t_i) \in \mathbb{R}$ with each $t_i \in S_1$, and such that:

$$t_i \in S \Rightarrow g(t_i) = f(t_i)$$

A branching rule B will be defined for any subset $\Gamma^k \subseteq S_1$ such that:

$$B(\Gamma^k) = \{\Gamma^{k+1}, \bar{\Gamma}^{k+1}\}$$

with:

$$\Gamma^k = \Gamma^{k+1} \cup \bar{\Gamma}^{k+1}$$

$$\text{and} \quad \Gamma^{k+1} \cap \bar{\Gamma}^{k+1} = \emptyset$$

On each of the subsets obtained by the application of B a bound will be defined as:

$$g^{k+1} = \begin{cases} \infty & \text{if no feasible solution exists} \\ \min\{g(t_i)/t_i \in \Gamma^{k+1}\} & \text{otherwise.} \end{cases}$$

and:

$$\bar{g}^{-k+1} = \begin{cases} \infty & \text{if no feasible solution exists} \\ \min\{g(t_i)/t_i \in \bar{\Gamma}^{-k+1}\} & \text{otherwise.} \end{cases}$$

For the purpose of this chapter, the term "no feasible solution exists" as used in the definition of the bounds is employed whenever one of the following conditions is encountered:

- 1) $\Gamma^{k+1} = \phi$ or $\bar{\Gamma}^{-k+1} = \phi$
- 2) $\Gamma^{k+1} \cap S = \phi$ or $\bar{\Gamma}^{-k+1} \cap S = \phi$
- 3) If, for some $R < k + 1$,

$$\min\{t_j/t_j \in \Gamma^{k+1}\} \geq \min\{t_j/t_j \in \Gamma^R\}$$

or,

$$\min\{t_j/t_j \in \bar{\Gamma}^{-k+1}\} \geq \min\{t_j/t_j \in \bar{\Gamma}^{-R}\}$$

- 4) If, for some $R < k + 1$,

$$\min\{t_j/t_j \in \bar{\Gamma}^{-k+1}\} \geq \min\{t_j/t_j \in \bar{\Gamma}^{-R}\}$$

or,

$$\min\{t_j/t_j \in \Gamma^{k+1}\} \geq \min\{t_j/t_j \in \Gamma^R\}$$

If any of the above conditions hold, then there is no further information to be gathered from the particular subset and, consequently, a bound of ∞ is assigned to it. Any subset which has assigned to it such a bound will be excluded from further consideration.

From a sequential application of \mathcal{B} , a contraction of S_1 is defined by:

$$S_1 \supset \Gamma^1 \supset \Gamma^2 \supset \Gamma^3 \supset \dots \supset \Gamma^P$$

with $\Gamma^i \neq \phi$ if $g^i < \infty$ for any $i \in \{1, 2, \dots, P\}$. The above contraction, obtained by the sequential application of \mathcal{B} , terminates when the last application of \mathcal{B} yields a subset Γ^P for which one of the following conclusions can be drawn:

- a) When an optimal solution $t_k \in \Gamma^P$ has been found, defined by $g(t_k) = \min\{g(t_i) / t_i \in \Gamma^P\}$, such that $t_k \in S$.
- b) When $g^P = \infty$.

If conclusion a) can be drawn, then further application of \mathcal{B} to Γ^P can yield no better results and, consequently, all subsets $\Gamma^i \subseteq \Gamma^P$ have been implicitly examined and can be discarded from further examination.

If conclusion b) can be drawn, then $\Gamma^P \cap S = \phi$ and, consequently, all subsets $\Gamma^i \subseteq \Gamma^P$ have been implicitly examined and can be discarded from further examination. Since \mathcal{B} is such that:

$$\Gamma^{P-1} = \Gamma^P \cup \bar{\Gamma}^P \quad \Gamma^P \cap \bar{\Gamma}^P = \phi$$

then, if Γ^P can be discarded from further examination, the subset Γ^{P-1} can be further contracted by the sequential application of \mathcal{B} starting with $\mathcal{B}(\bar{\Gamma}^P)$. This process shall be referred to as backtracking. The sequential application of \mathcal{B} is again terminated when conclusion a) or b) can be drawn.

Since S_1 is defined to be a finite set, the number of subsets that can be generated in a contraction by the successive application of \mathcal{B} is finite. Furthermore, since any subset Γ^i or $\bar{\Gamma}^i$ is finite, the number of subsets that can be generated in a contraction on any subset is also finite.

Any contraction of a subset must terminate. Consider the case in which the last pair of non-empty subsets that can be generated by the application of \mathcal{B} , say Γ^q and $\bar{\Gamma}^q$, each contain only one element, i.e.:

$$\Gamma^q = \{t_r\}$$

$$\bar{\Gamma}^q = \{t_s\}$$

in that case, if $t_r \in S$, then conclusion a) can be drawn; otherwise, conclusion b) is drawn by the definition of g^P . Consequently, subset Γ^q will be discarded from consideration. Since $\Gamma^{q-1} = \Gamma^q \cup \bar{\Gamma}^q$, Γ^{q-1} can be examined by only considering $\bar{\Gamma}^q$. Again, if $t_s \in S$, then conclusion a) holds, otherwise conclusion b) holds. In either case, subset $\bar{\Gamma}^q$ is discarded from further examination, and immediately subset Γ^{q-1} is also discarded from further consideration. The number of unexplored subsets

remaining has thus been reduced.

Consider now the case in which, for some subset Γ^q containing more than one element, conclusion b) can be drawn. In that case the subset Γ^q is immediately discarded from further consideration. On the other hand, if conclusion a) can be drawn, then the optimal element contained in Γ^q has been found, and no solution contained in that subset will be better in terms of the function f . Consequently, the subset Γ^q can be discarded from further consideration. In either case, the number of subsets that still must be considered has been reduced.

The above observations may be summarized by stating that an algorithm based on the presented principles is finite and terminates with an optimal solution, or with the conclusion that $S = \phi$. Finiteness is a result of the properties of the branching rule \mathcal{B} which creates contractions on subsets of the superset S_1 . A contraction is terminated when conditions a) or b) are met. The backtracking process insures that no subsets are excluded from either implicit or explicit consideration and that any subset already considered will not be considered again.

The Algorithm

The problem to be considered in this thesis is of the form:

$$\text{minimize} \quad C^T X$$

subject to:

$$A_1 X = b_1$$

$$A_2 X \leq b_2$$

X = Vector with binary components.

where:

A_1 - m_1 by n matrix possessing the unimodular property.

A_2 - m_2 by n matrix with integer components.

C - n vector of cost coefficients.

b_1 and b_2 - m_1 and m_2 component vectors with integer components, respectively.

Defining:

$$S_1 = \{X/A_1 X = b_1, X = \text{Vector with binary components}\}$$

$$S = \{X/X \in S_1 \text{ and } A_2 X \leq b_2\}$$

Then the best solution value contained in the superset S_1 is obtained by solving:

$$\text{minimize} \quad C^T X$$

subject to:

$$A_1 X = b_1$$

$$X = \text{binary}$$

denoting the optimal solution to the above problem by X^* then, if $X^* \in S$,

the optimal solution to the entire problem has been obtained and the algorithm is terminated. Otherwise the algorithm proceeds to construct a feasible solution.

In order to construct a feasible solution to the problem, the set S_1 will be partitioned into two mutually exclusive subsets. This can be accomplished by noting that, in any solution, x_j must assume a value of zero or one. The initial partition of S_1 will be made by the rule:

$$S_1 = S_1^+ \cup S_1^- \text{ with } S^+ \cap S^- = \phi$$

where:

$$S_1^+ = \{X = [1, x_2, x_3, \dots, x_n] / X \in S_1\}$$

$$S_1^- = \{X = [0, x_2, x_3, \dots, x_n] / X \in S_1\}.$$

Each vector in each of the subsets contains the variable x_1 "fixed" at a specific value. All variables not "fixed" will be referred to as "free" variables.

The constraint set $A_2 x \leq b_2$ may be rewritten as:

$$\sum_{j=1}^n [A_2]_{ij} x_j \leq b_{2i} \quad i=1, m_2$$

where $[A_2]_{ij}$ refers to the entry in the i th row and j th column of the matrix A_2 , and b_{2i} as the i th entry in the vector b_2 . This constraint set may be partitioned in terms of the present "fixed" and "free" variables in the manner:

$$\sum_{j=2}^n [A_2]_{ij} x_j \leq b_{2i}^1$$

where:

$$b_{2i}^1 = b_{2i} - [A_2]_{i1} x_1$$

Since for any $X \in S$ the above constraint must be satisfied, it is possible to derive various tests in order to determine if and under what conditions it is possible to obtain a feasible solution to the problem. For the above additional constraint representation consider, for example, all constraints such that $b_{2i}^1 = 0$. Without loss of generality, consider constraint k as one of these. Then, if $[A_2]_{kj} \geq 0$ for $j=2, n$, then any solution which is feasible must meet the following conditions:

- (1) $x_j = 0$ if $[A_2]_{kj} > 0$.
- (2) $x_j = \text{"free"}$ if $[A_2]_{kj} = 0$.

Consequently, it is possible to consider all the variables which meet condition (1) as implicitly fixed as a result of fixing variable x_1 .

Consider another situation in which for a given constraint:

$$B = \sum_{j=2}^n \min\{0, [A_2]_{ij}\}$$

Then, if $b_{2i}^1 < 0$ and $B > b_{2i}^1$, no feasible solution exists to the problem with the present set of fixed variables.

Depending on the properties of the coefficient matrix A_2 , more specialized tests can be devised. For example, if $A_2 \geq 0$, then B will always assume a value of zero, and as soon as b_{2i}^1 assumes a negative value, the conclusion is drawn that no feasible solution exists to the problem for the particular set of fixed variables.

Based on the results presented so far, it has been shown that the original superset S_1 of the solution space S can be partitioned into two mutually exclusive subsets S_1^+ and S_1^- such that $S_1^+ \cup S_1^- = S_1$. Furthermore, it was shown that through the use of various tests on the additional constraints, indications can be obtained as to whether either of these subsets do not contain feasible solutions to the problem. If the tests indicated that some variables are implicitly fixed, then these variables are fixed at their values.

Partitioning the imbedded problem according to fixed and free variables, a lower bound on the solutions contained in either set may be obtained by solving the problems:

For the set S_1^+

$$\text{minimize} \quad C_{N^+}^T X_{N^+} + C_{F^+}^T X_{F^+}$$

subject to:

$$A_{1N^+} X_{N^+} = b_1 - A_{1F^+} X_{F^+}$$

X_{N^+} = Vector with binary components

where:

C_{N^+} - Cost coefficient vector of free variables associated with the subset S_1^+ .

X_{N^+} - Vector of free variables associated with the subset S_1^+ .

A_{1N^+} - Constraint coefficient matrix associated with free variables of the subset S_1^+ .

C_{F^+} - Cost coefficient vector of fixed variables associated with the subset S_1^+ .

X_{F^+} - Vector of fixed variables associated with the subset S_1^+ .

A_{1F^+} - Constraint coefficient matrix associated with fixed variables of subset S_1^+ .

Denoting $X_{N^+}^*$ as the optimal solution to the above problem, a lower bound on the solutions contained in S_1^+ is given by:

$$g_1^+ = \begin{cases} \infty & \text{if no feasible solution exists to} \\ & \text{imbedded problem or additional} \\ & \text{constraints with present set of} \\ & \text{fixed variables} \\ C_{N^+}^T X_{N^+}^* + C_{F^+}^T X_{F^+} & \text{otherwise} \end{cases}$$

By similar means, a lower bound on the solutions contained in S_1^- may be obtained by solving the problem.

$$\text{minimize} \quad C_{N^-}^T X_{N^-} + C_{F^-}^T X_{F^-}$$

subject to:

$$A_{N^-} X_{N^-} = b_1 - A_{F^-} X_{F^-}$$

X_{N^-} = Vector with binary components

where the notation is the same as above, except the minus superscripts indicate the relation to the subset S_1^- .

A lower bound on the solutions contained in S_1^- is given in a similar manner, i.e.:

$$g_1^- = \begin{cases} \infty & \text{if no feasible solution} \\ & \text{exist to the imbedded} \\ & \text{problem or the additional} \\ & \text{constraints with the present} \\ & \text{set of fixed variables.} \\ C_{N^-}^T X_{N^-}^* + C_{F^-}^T X_{F^-} & \text{otherwise} \end{cases}$$

The above lower bounding problems are amenable to efficient solution procedures since the constraint coefficient matrices A_{1N^+} and A_{1N^-} still possess the unimodular property. What has been accomplished so far is summarized in Figure 1.

As a result of the lower bound problem solutions, it is possible to have obtained a feasible or optimal solution to the entire problem at this level of computation. An optimal solution has been obtained if one of the following conditions is satisfied:

if (1) $g_1^- \neq \infty$

(2) $g_1^- \leq g_1^+$

$$(3) \quad A_2 X \leq b_2 \quad \text{where} \quad X = [X_{F+}, X_{N+}^*]^T.$$

then X is an optimal solution to the problem.

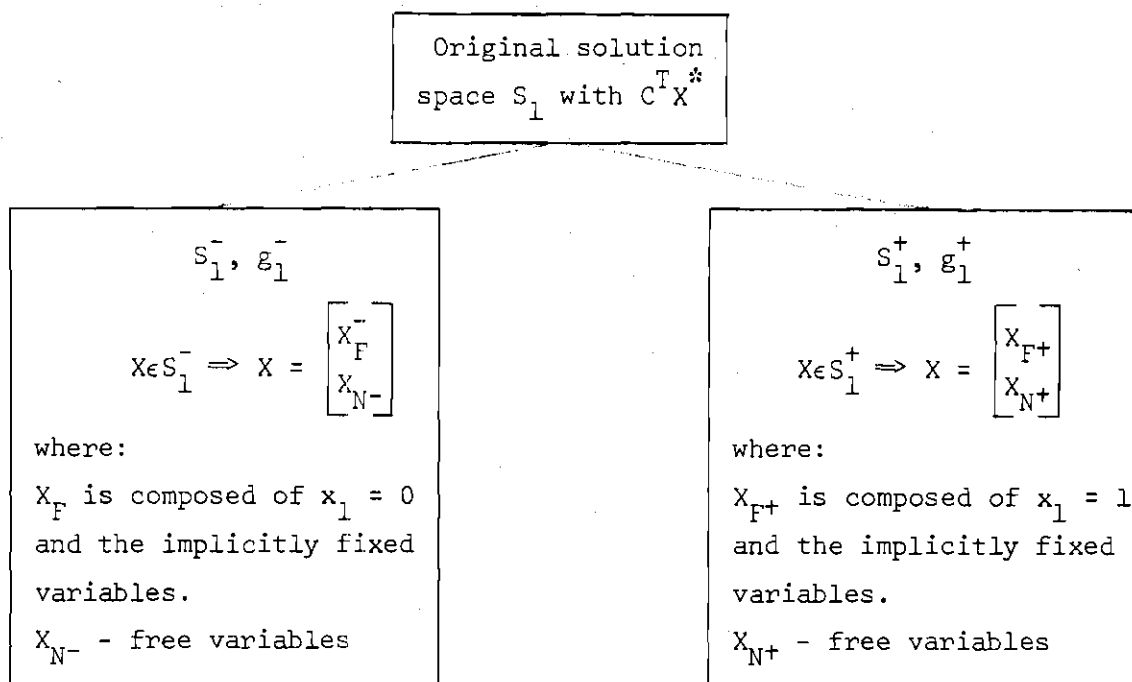


Figure 1. Summary Results from Initial Steps of Algorithm

Condition 2

If (1) $g_1^+ \neq \infty$

(2) $g_1^+ \leq g_1^-$

(3) $A_2 X \leq b_2$ where $X = [X_{F+}, X_{N+}^*]^T$

then X is an optimal solution to the problem. A feasible solution has been obtained which, if optimal, cannot yet be identified as such when one of the following conditions holds:

Condition 1a

If (1) $g_1^- \neq \infty$

(2) $g_1^- > g_1^+$

(3) $A_2 X \leq b_2$ where $X = [X_{F-}, X_{N-}^*]^T$

then X is a feasible solution to the problem.

Condition 2a

If (1) $g_1^+ \neq \infty$

(2) $g_1^+ > g_1^-$

(3) $A_2 X \leq b_2$ where $X = [X_{F+}, X_{N+}]^T$

then X is a feasible solution to the problem.

If condition 1a holds, then the subset S_1^- does not need to be considered further since X is an optimal solution contained in the set. To denote that no further search in the set is required, the feasible solution is stored and its objective function value is stored in the variable v . Furthermore, the lower bound on the subset is changed to ∞ . The variable v will always contain the objective function value of the best feasible solution found so far.

If condition 2a holds, then the same procedure as above is performed, except all conclusions must be applied to the subset S_1^+ .

Whenever a feasible solution is referred to in the rest of this chapter, it is meant a feasible solution which is better in objective function value than the best one found so far.

It is also possible to arrive at conclusions regarding the exclusion of any one or both of the subsets from further search for a

feasible solution. Consider the following:

Condition 3

If (1) $g_1^- = \infty$

(2) $g_1^+ = \infty$

then no feasible solution exists to the problem.

Condition 4

If (1) $g_1^- = \infty$

then no feasible solution exists in the subset S_1^- .

Condition 4a

If (1) $g_1^+ = \infty$

then no feasible solution exists in the subset S_1^+ .

It should be noted that condition 3 can hold even if the initial imbedded problem yields a feasible solution. Conditions 4 and 4a hold by definition of g_1^- and g_1^+ , respectively.

The computations associated with this level are continued based on the following rules:

- a) If condition 4 holds, or $g_1^+ \leq g_1^-$,
then select S_1^+ for further partitioning
- b) If condition 4a holds, or $g_1^- < g_1^+$,
then select S_1^- for further partitioning.

The above rules are based on the hypothesis that, if the subset with the lowest lower bound is selected for further partitioning, it is likely, based on the information available, that it contains a feasible solution which has a lower objective function value than the other subset.

At this point the computations associated with the beginning of the algorithm are finished. Before proceeding to show the computations associated with some intermediate level of computation, it will be convenient to establish a more general notation. Let:

Γ^i - Subset selected for further partitioning at the i th level of computation. At the first level, Γ^1 is selected based on rules a) and b). Γ^0 is S_1 .

$\bar{\Gamma}^i$ - Subset not selected for further partitioning at the i th level of computation. At the first stage, $\bar{\Gamma}^1$ is selected according to rules a) and b).

g^i - Lower bound on subset Γ^i .

\bar{g}^i - Lower bound on subset $\bar{\Gamma}^i$.

X_F^i - Vector of fixed variables at termination of the i th level of computation.

X_N^i - Vector of free variables at termination of the i th level of computation.

Any subset for which the lower bound is ∞ will be considered as inactive in the sense that no additional feasible solutions are contained in that set. Any subset not considered inactive will be denoted as active in the sense that it may contain feasible solutions which have not yet been identified.

The computations associated with some intermediate level of computation, say the k th level, would involve the following:

A. Branching

Branching refers to the partitioning of the subset Γ^T selected at the previous level of computation. Recalling that:

$$\Gamma^K = \{X = [X_F^{K-1}, X_N^{K-1}]^T / X \in S_1\}$$

where:

$$X_F^{K-1} = [\bar{x}_{(1)} \quad \bar{x}_{(2)} \quad \cdots \quad \bar{x}_{(r)}]^T$$

$$X_N^{K-1} = [x_{(r+1)} \quad x_{(r+2)} \quad \cdots \quad x_{(n)}]^T$$

The bars over the components of X_F^{K-1} serve to indicate that they are the values of the variables that are fixed. The components of X_N^{K-1} are the free variables. The subscripts enclosed in the parenthesis serve to indicate the order in which the variables are fixed.

The branching rule B applied to the subset Γ^+ will have the effect:

$$B(\Gamma^K) = \{S_K^+, S_K^-\}$$

where:

$$S_K^+ = \{X = [Y^+, N]^T / X \in S_1\}$$

$$S_K^- = \{X = [Y^-, N]^T / X \in S_1\}$$

$$Y^+ = [\bar{x}_{(1)}, \bar{x}_{(2)}, \dots, \bar{x}_{(r)}, x_{(r+1)}=1]^T$$

$$N = [x_{(r+2)}, x_{(r+3)}, \dots, x_{(n)}]^T$$

$$Y^- = [\bar{x}_{(1)}, \bar{x}_{(2)}, \dots, \bar{x}_{(r)}, x_{(r+1)} = 0]^T$$

As can be noted, the branching rule \mathcal{B} provides a partitioning of the subset Γ^K in the sense that:

$$S_K^+ \cup S_K^- = \Gamma^K$$

$$S_K^+ \cap S_K^- = \phi$$

The selection of the element $x_{(r+1)}$ at any level can be made according to many rules. Examples of some of these are:

- a. Select the first element from the vector of free variables as $x_{(r+1)}$.
- b. Select $x_{(r+1)}$ such as to maximize $|g^K - g^{K-1}|$. This can be accomplished by considering the dual evaluators associated with the solution of the lower bounding problem from the previous level of computation.
- c. Select $x_{(r+1)}$ so as to achieve maximum reduction of some measure of infeasibility associated with the additional constraints.

As an example of a measure of infeasibility of the additional constraints, consider the value criterion used by Balas [3] for augmenting a partial solution, namely define:

$$I = \sum_{i \in M_j} (b_{2i} - \sum_{j \in \bar{\theta}} [A_2]_{ij} \bar{x}_j - [A_2]_{iL})$$

where:

$\bar{\theta}$ - Index set of fixed variables.

\bar{x}_j - Value at which x_j is fixed.

θ - Index set of free variables

$L \in \theta$.

$$M_j = \{i / (b_{2i} - \sum_{j \in \bar{\theta}} [A_2]_{ij} \bar{x}_j - [A_{2L}]) < 0\}.$$

The variable to consider next for the branching rule would be x_{L^*} , where L^* is the index of the variable which maximizes I .

B. Bounding

After the branching rule has been applied, two mutually exclusive subsets must be considered for further branching. This algorithm makes this selection based on the values obtained for the lower bounds on the hypothesis that the subset with the lowest lower bound is more likely to contain the optimal solution than the other. Consequently, it is desirable to obtain "tight" lower bounds. In order to obtain the "tightest" lower bounds on each of the subsets, the additional constraints should be examined in order to find any variables which are implicitly fixed through the explicit fixing of $x_{(r+1)}$. These may be determined by consecutively considering first a partition of the additional constraints:

$$\left. \begin{aligned} \sum_{j=r+2}^n [A_2]_{i(j)} x_{(j)} &\leq b_{2i} - \sum_{j=1}^r [A_2]_{i(j)} \bar{x}_j - [A_2]_{i(r+1)} \\ \text{and next:} \\ \sum_{j=r+2}^n [A_2]_{i(j)} x_{(j)} &\leq b_{2i} - \sum_{j=1}^r [A_2]_{i(j)} \bar{x}_j \end{aligned} \right\} i = 1, m_2$$

in either case denote the right-hand side by b_{2i}^K . A general test that can be made to determine the implicitly fixed variables is:

For every constraint satisfying:

$$a) \quad b_{2i}^K = 0$$

$$b) \quad [A_2]_{i(j)} \geq 0 \quad j=r+2, n$$

set $x_j = 0$ if $[A_2]_{i(j)} > 0$, i.e., x_j is implicitly fixed at a value of zero.

In some instances, information can also be gained from the additional constraints as to whether it is not possible to obtain a feasible solution to the problem with the present set of fixed variables. A general test that may be used to determine this was developed by Balas [3], and is given below:

If, for any constraint, $b_{2i}^K < 0$ and

$$\sum_{j=r+2}^n \min\{0, [A_2]_{i(j)}\} > b_{2i}^K$$

then no feasible solution exists to the problem with the present set of fixed variables.

denoting the vector of explicitly and implicitly fixed variables associated with the set S_K^+ by γ^+ and the vector of free variables associated with this set by ϕ^+ , then a lower bound on the solutions contained in S_K^+ is obtained by solving the problem:

$$\text{minimize} \quad C_{\phi^+}^T \phi^+ + C_{\gamma^+}^T \gamma^+$$

subject to:

$$A_{1\phi^+} \phi^+ = b_1 - A_{1\gamma^+} \gamma^+$$

ϕ^+ = Vector with binary components

where:

C_{ϕ^+} - Vector of cost coefficients associated with the free variables.

C_{γ^+} - Vector of cost coefficients associated with fixed variables.

$A_{1\phi^+}$ - Coefficient matrix of free variables associated with imbedded problem.

$A_{1\gamma^+}$ - Coefficient matrix of fixed variables associated with imbedded problem.

denoting the optimal solution to the above problem by $[\gamma^+ \phi^{+*}]^T$, then the lower bound on the subset S_K^+ is given by:

$$g_K^+ = \begin{cases} \infty & \text{if } [C_{\phi^+}^T C_{\gamma^+}^T] \begin{bmatrix} \phi^{+*} \\ \gamma^+ \end{bmatrix} \geq v \\ \infty & \text{if no feasible solution to above} \\ & \text{problem, or test indicated no feasible} \\ & \text{solution to additional constraints.} \\ [C_{\phi^+}^T C_{\gamma^+}^T] \begin{bmatrix} \phi^{+*} \\ \gamma^+ \end{bmatrix} & \text{otherwise.} \end{cases}$$

Furthermore, if $g_K^+ < \infty$ and $A_2 [\gamma^+ \phi^{+*}]^T \leq b_2$, then a feasible solution

has been found. Consequently it is stored and the following changes are made:

$$v = g_K^+$$

$$g_K^+ = \infty$$

Denoting the vector of implicitly and explicitly fixed variables associated with S_K^- by γ^- and the vector of free variables associated with this set by ϕ^- , a lower bound on S_K^- can be obtained by solving the problem:

$$\text{minimize} \quad C_{\phi^-}^T \phi^- + C_{\gamma^-}^T \gamma^-$$

subject to:

$$A_{1\phi^-} \phi^- = b_1 - A_{1\gamma^-} \gamma^-$$

ϕ^- = Vector with binary components

where the definition of the terms is the same as with the previous problems, except that the superscript indicates that the problem is associated with the subset S_K^- .

Denoting the optimal solution to the above problem by $[\gamma^- \phi^{-*}]^T$, then the lower bound on the subset S_K^- is given by:

$$g_K^- = \begin{cases} \infty & \text{if } [C_{\phi}^T \ C_{\gamma}^T] \begin{bmatrix} \phi^- \\ \gamma^- \end{bmatrix} \geq v \\ \infty & \text{if no feasible solution to above problem,} \\ & \text{or test indicated no feasible solution to} \\ & \text{additional constraints} \\ [C_{\phi}^T \ C_{\gamma}^T] \begin{bmatrix} \phi^- \\ \gamma^- \end{bmatrix} & \text{otherwise} \end{cases}$$

Furthermore, if $A_2[\gamma^- \ \phi^-]^T \leq b_2$ and $g_K^- < \infty$, then a feasible solution has been found. Consequently it is stored and the following changes are made:

$$v = g_K^-$$

$$g_K^- = \infty$$

It should be noted that, if at any time, $v = CX^*$, then the feasible solution associated with v is optimal. In such cases, the optimal solution to the problem is an alternate optima of the imbedded problem. In that case the algorithm is terminated.

C. Choice of Subset for Further Partitioning

Before explicitly stating the choice rule to select a subset for further partitioning, define the following:

$$\text{If } g_K^+ \leq g_K^-$$

Then:

X_F^K = Vector of explicitly and implicitly fixed variables associated with subset S_K^+ .

X_N^K = Vector of free variables associated with subset S_K^+ .

$\Gamma^{K+1} = \{X = [X_F^K, X_N^K]^T / X \in S_1\}$.

$g^{K+1} = g_K^+$.

\bar{X}_F^K = Vector of explicitly and implicitly fixed variables associated with subset S_K^- .

\bar{X}_N^K = Vector of free variables associated with subset S_K^- .

$\bar{\Gamma}^{K+1} = \{X = [\bar{X}_F^K, \bar{X}_N^K]^T / X \in S_1\}$.

$g^{-K+1} = g_K^-$.

If $g_K^- < g_K^+$. Then:

X_F^K = Vector of implicitly and explicitly fixed variables associated with subset S_K^- .

X_N^K = Vector of free variables associated with subset S_K^- .

$\Gamma^{K+1} = \{X = [X_F^K, X_N^K] / X \in S_1\}$.

$g^{K+1} = g_K^-$.

\bar{X}_F^K = Vector of implicitly and explicitly fixed variables associated with subset S_K^+ .

\bar{X}_N^K = Vector of free variables associated with subset S_K^+ .

$\bar{\Gamma}^{K+1} = \{X = [\bar{X}_F^K, \bar{X}_N^K] / X \in S_1\}$.

$g^{-K+1} = g_K^+$.

With the above definition, the choice rule for the algorithm will be:

If $g^{K+1} = \infty$, then go to section D;

otherwise select Γ^{K+1} for further partitioning.

It should be noted that in the first case, if $g^{K+1} = \infty \Rightarrow \bar{g}^{K+1} = \infty$ and hence, by a previous definition, both of the subsets Γ^{K+1} and $\bar{\Gamma}^{K+1}$ are inactive.

It should also be noted that the above is not the only choice rule that can be employed. Examples of other choice rules that could be employed are:

a) Select for further partitioning the subset Γ^y , where Γ^y is such that y is the index of the subset which satisfies:

$$\min\left\{ \min_{g^i < \infty} \{g^i, i=1, K+1\}, \min_{\bar{g}^i < \infty} \{\bar{g}^i, i=1, K+1\} \right\}$$

In other words, select for partitioning the active subset with the least lower bound.

Even though the above choice rule is appealing, its principal drawback for the problems considered was the time required to obtain a feasible solution. Computational experience with this choice rule for the algorithm presented in this chapter has proven it to be impractical.

b) Another choice rule that can be employed is to select for partitioning the first subset for which $g^i < \infty$ or $\bar{g}^i < \infty$. In other words, no attention is paid to the lower bounds except when infeasibility is encountered. It should be recalled that infeasibility denotes either infeasibility with respect to the constraint sets, or that it can be concluded that a particular subset does not contain a feasible solution with objective function value less than v .

The choice rule that has been selected for the algorithm is, in

a sense, a compromise between a) and b) in that the subset selected for further partitioning is the one which has the least lower bound at the given level of computation.

D. Backtracking

The sequence of computations to be described in this section is carried out only after more than one level of computation has been performed. Consider that at the k th level of computation the test described in part c) indicates that the computations described in this section are to be carried out. At that point, the results of the previous computations can be described graphically as in Figure 2. It should also be recalled that the partitioning scheme has the property:

$$\begin{aligned}
 \Gamma^0 &= \bar{\Gamma}^1 \cup \Gamma^1, & \bar{\Gamma}^1 \cap \Gamma^1 &= \phi \\
 \Gamma^1 &= \bar{\Gamma}^2 \cup \Gamma^2, & \bar{\Gamma}^2 \cap \Gamma^2 &= \phi \\
 \Gamma^2 &= \bar{\Gamma}^3 \cup \Gamma^3, & \bar{\Gamma}^3 \cap \Gamma^3 &= \phi \\
 &\vdots & &\vdots \\
 \Gamma^K &= \bar{\Gamma}^{K+1} \cup \Gamma^{K+1}, & \bar{\Gamma}^{K+1} \cap \Gamma^{K+1} &= \phi \\
 \Rightarrow \Gamma^i &\subset \Gamma^{i-1} & i=1, K+1
 \end{aligned}$$

Now, since both Γ^{K+1} and $\bar{\Gamma}^{K+1}$ are inactive, the implication is that Γ^K is also inactive since $\Gamma^K = \Gamma^{K+1} \cup \bar{\Gamma}^{K+1}$ and $\Gamma^{K+1} \cap \bar{\Gamma}^{K+1} = \phi$. To denote that Γ^K is inactive, let $g_K = \infty$. Before proceeding, the information associated with Γ^{K+1} and $\bar{\Gamma}^{K+1}$ is discarded. The computations are continued as follows:

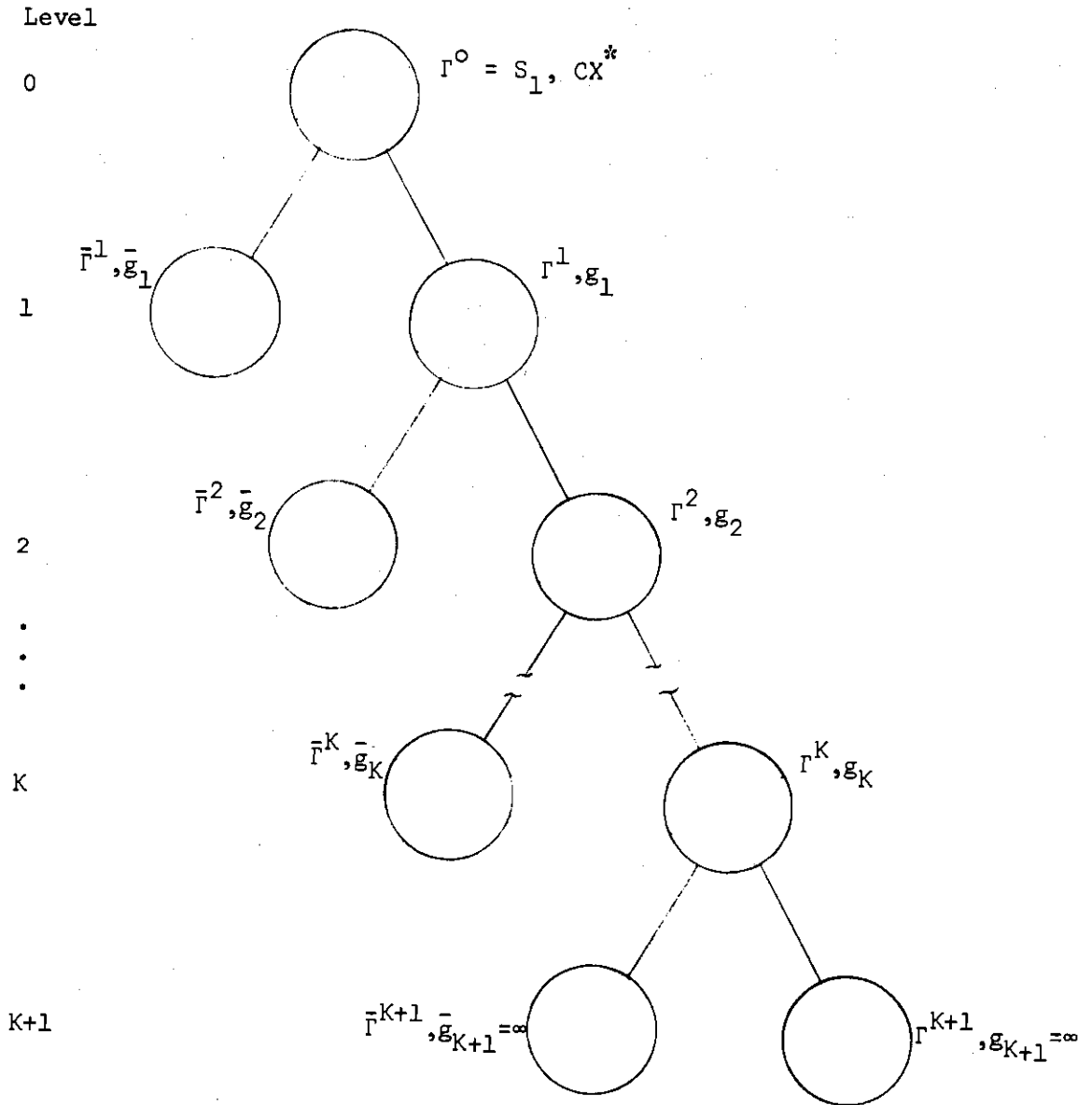


Figure 2. General Results of Algorithm at Some Intermediate Level of Computation

(1) If $\bar{g}_K < v$, then let:

$$\text{new } \Gamma^K = \text{old } \bar{\Gamma}^K$$

$$\text{new } g_K = \text{old } \bar{g}_K$$

and, for the original Γ^K and g_K ,

Let:

$$\text{old } \bar{\Gamma}^K = \text{old } \Gamma^K$$

$$\text{old } \bar{g}_K = \text{old } g_K$$

Continue with computations described in Part A.

If $\bar{g}_K \geq v$ then both Γ^K and $\bar{\Gamma}^K$ are inactive and, consequently, Γ^{K-1} is also inactive. Let $g^{K-1} = \infty$ to denote that subset Γ^{K-1} is inactive. Letting $K = K-1$, the computations starting at (1) above are repeated unless $K = 0$. When $K = 0$ there are no remaining active subsets and the computations are terminated.

Summary of Algorithm

The algorithm presented in the previous section of this chapter was constructed based on the general principles presented in the initial section of the chapter. Figure 3 is a graphical interpretation of the logical process described by the algorithm.

In Figure 3, the blocks labelled A1 through A4 denote the steps described in the branching part of the algorithm. The blocks labelled B1 through B8 represent the steps described in the bounding section of the algorithm. The block denoted by C1 represents the steps described in the section of the algorithm denoted by choice of subset for further partitioning. The blocks denoted by D1 through D3 represent the steps described in the section of the algorithm denoted by backtracking.

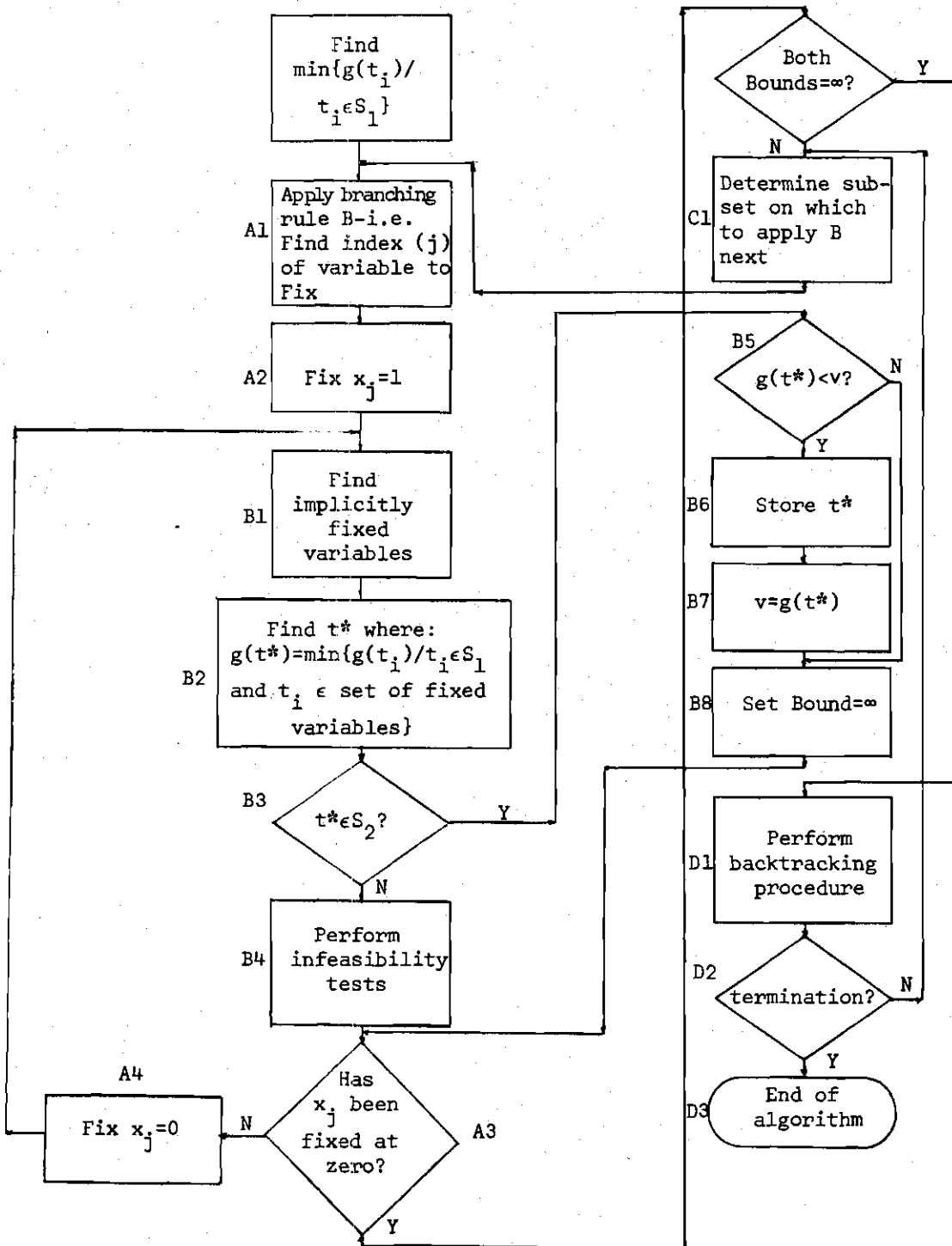


Figure 3. Summary Presentation of Algorithm

The correspondence between the algorithm and the basic principles presented is rather straightforward. The finite set S referred to in the initial section of the chapter is the set S defined for the algorithm, and the superset S_1 referred to in the basic principles is the set S_1 defined in the algorithm. Correspondingly, the function $f(x) = C^T X, X \in S$ and $g(X) = C^T X, X \in S_1$.

The branching rule \mathcal{B} has been defined under part A of the algorithm, and the definitions of the subsets Γ^P and $\bar{\Gamma}^P$ obtained by \mathcal{B} are based on the results of part B and C of the algorithm, namely, the results of the bounding problems. It should be noted that the bounding problems are such that efficient computational procedures exist for their solution since their constraint coefficient matrices always retain the unimodular property. For bounding problems which do not have a special structure, linear programming methodology can be used, and for problems which exhibit some special structure for which more efficient solution algorithms are available, the particular algorithms applicable can be used.

Finally, the exclusion of the subsets from further consideration resulting from the concept of backtracking is presented in part D.

CHAPTER III
APPLICATION-FORMULATION OF
A COURSE SCHEDULING PROBLEM

This and the following chapter will present the application of the solution methodology developed in Chapter II to a course-time scheduling problem. The particular problem arose as part of an integrated management information and decision system developed for the School of Industrial and Systems Engineering of the Georgia Institute of Technology. In order to provide a context for the application, the next section will present the basic concepts of this system.

ISEMIS

The Industrial and Systems Engineering Management Information System (ISEMIS) is designed to provide rapid access to information pertaining to students enrolled in the School of Industrial and Systems Engineering. It also provides the administration of the school with both the methodology and computer programs necessary to make decisions regarding the courses that should be offered, the faculty that should be assigned to teach these courses, and the scheduling of the courses to times of the day. A conceptual presentation of ISEMIS is given in Figure 4.

The data base contains pertinent information concerning students of both academic and personal nature. In addition, it contains

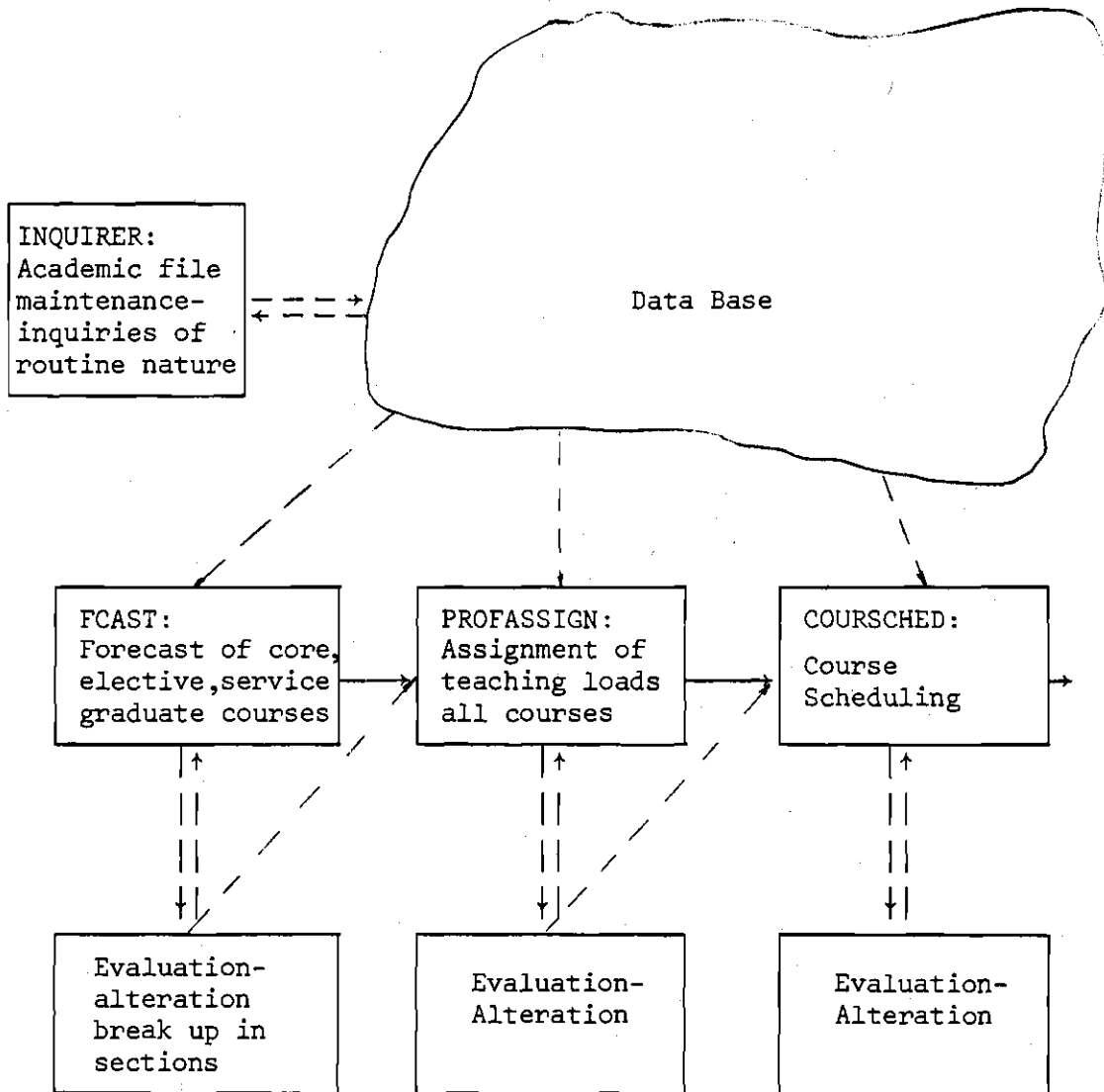


Figure 4. Conceptual Representation of ISEMIS

historical information concerning past course enrollments, information concerning faculty teaching preferences as well as maximum allowable teaching loads.

Four separate subsystems draw information from the data base. These are the Inquirer, Fcast, Profassign, and Coursched subsystems. The Inquirer subsystem consists of a set of programs operable in either a timesharing or batch mode, which allow speedy and efficient file maintenance and tabulation of statistics concerning groups of students or individual students.

The Fcast subsystem provides forecasts of the number of students that will have demand for given courses. The courses are subdivided into required courses, and elective or service courses. For the required courses a forecast is made based on the results obtained from checking the course information of all students contained in the data base. For elective or service courses, a forecast is made based on a statistical analysis of the past enrollments. In either case, the effectiveness of the forecasting system is monitored on a continuing basis in order to detect when changes have to be made to the model in order to obtain better forecasts.

From the results of the Fcast subsystem, the school administration must make a decision regarding how many sections, if any, of a given course should be offered. Once this information is made available, the Profassign subsystem may be used to assign faculty to teach these courses. The actual assignment of courses to faculty is made so as to minimize the sum of the costs associated with a professor

teaching a given course. For each professor and course, the associated cost is a function of the desire the professor has to teach that course, and the degree of competence he has to teach it, as decided by the administration of the school. Constraints on this minimization problem consist of the maximum and minimum teaching loads permissible for each professor; again, as determined by the School Administration based on its knowledge of the other commitments a faculty member has such as research, student advising, and membership on various committees.

The results obtained from the Fcast and Profassign subsystems yield a listing of courses to be offered together with the faculty assigned to teach these courses. With this information, together with a knowledge of the physical facilities available and the courses which, if taught simultaneously, would create conflicts for students or faculty or both, the Coursched subsystem can create a schedule indicating the times various courses should be offered. Since the problems created within this subsystem will be used as application areas for the algorithm developed in the previous chapter, this subsystem will be explained in greater detail in the next section of this chapter.

The Coursched Subsystem

The basic purpose of the Coursched Subsystem is to provide an answer to the question, "Given a set of courses which will be taught and a list of teaching assignments, at what times of the day should these courses be offered?" The answer to this question is provided subject to the following information being available:

1. A relative preference rating indicating what the most desirable times are to offer courses.
2. The number of classrooms available and their student capacities.
3. A list of courses which, if taught simultaneously, will create conflicts for students, or faculty, or both.

With the above information, a mathematical model can be constructed which, when solved, will yield a schedule satisfying the following restrictions:

1. All courses taught by the same faculty member are offered at different times.
2. There will be enough classrooms, if available, of the appropriate size to accommodate all courses offered at any one time.
3. No more than one course required by the same student type will be taught at the same time.
4. All courses will be scheduled if possible.

A schematic representation of the information required and generated by the Coursched subsystem in relationship to ISEMIS is provided in Figure 5. In the next section of this chapter, the mathematical model used to represent the course scheduling problem will be developed in detail. In subsequent sections, the properties of the model will be presented.

Mathematical Formulation

In general it will be assumed that there are m possible time slots at which any one of n courses could possibly be assigned. The basic decision to be made then is whether course j ($j=1,2,\dots,n$) is to

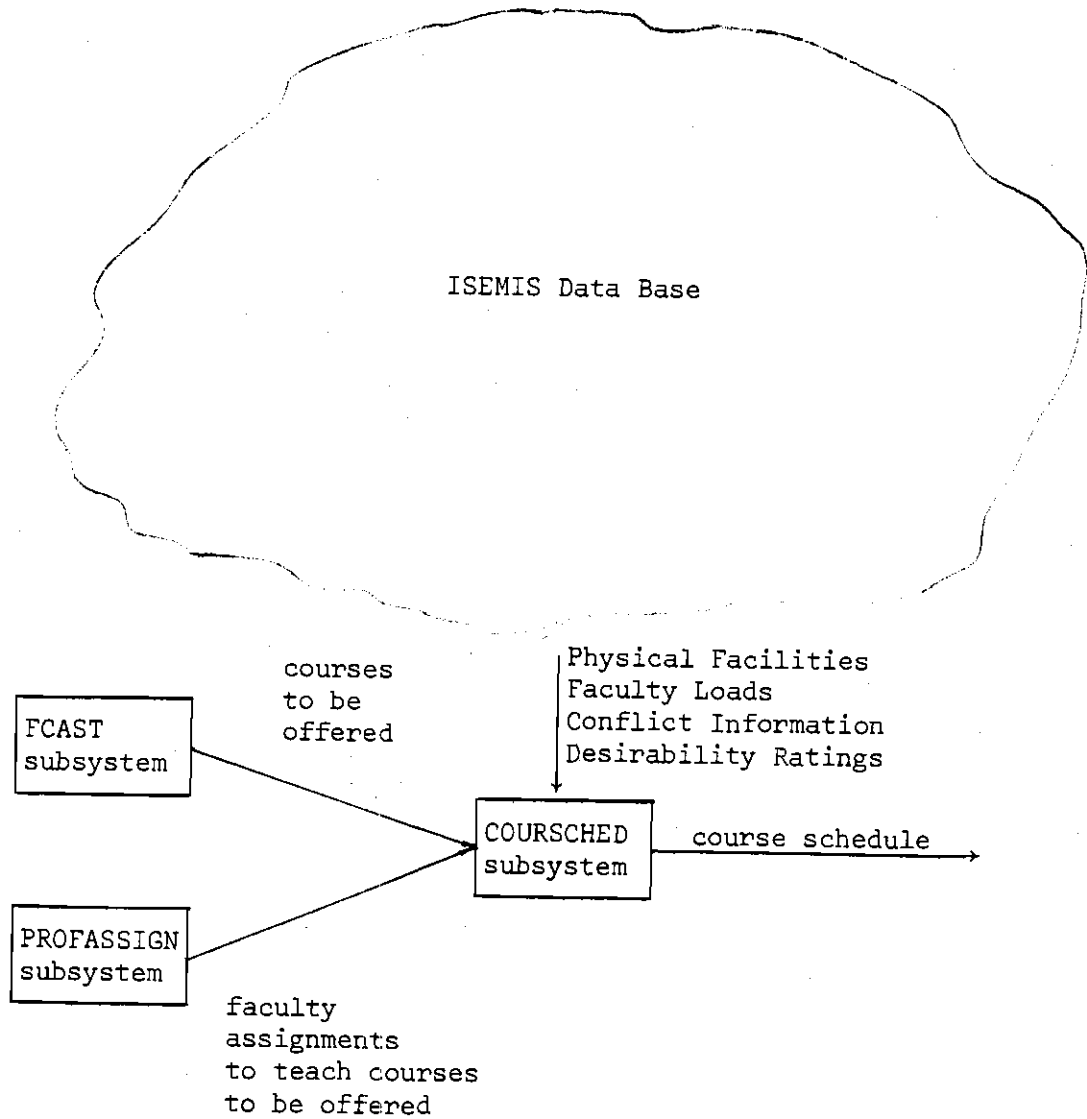


Figure 5. Coursched in Relation to ISEMIS

be offered at time $i(i=1,2,\dots,m)$. As such, the decision variable will be x_{ij} , where:

$$x_{ij} = \begin{cases} 1 & \text{if course } j \text{ is to be taught at time } i \\ 0 & \text{otherwise.} \end{cases}$$

Based on information available to the school administration regarding what time slots are better than others in which to offer a given course j , it is possible to assign a set of preference numbers to each course for each time. It will be assumed that the lower the preference number c_{ij} , the more preferred it is to offer course j at time i . It is also assumed that the objective of the school administration will be to offer a schedule which has the lowest total preference measurement. In other words, the objective to be pursued is to minimize the expression:

$$\sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij}$$

the constraints imposed on this minimization fall into four categories. The constraints in category one must insure that the solution to the model has indeed scheduled all courses that must be offered at some time. This may be expressed in the following manner:

$$\sum_{i=1}^m x_{ij} = 1 \quad j=1,2,\dots,n$$

Category two constraints are those expressing the limitations on available classroom space at given times. In most situations, classrooms

available in a given school can be subdivided into groups, each having a maximum number of seats.

Supposing that there are m_1 groupings of classrooms, each of which has a different maximum student capacity, let N_i^k denote the number of classrooms available in category k ($k=1,2,\dots,m$) at time i . The subscript i is retained to acknowledge that in many instances various classrooms are reserved at specific times for faculty, special group or committee meetings; hence, the number of classrooms of a certain group available to accommodate courses is variable. Denoting the maximum student capacity for classrooms in the k th group by U^k , and ordering the k 's in such a manner that $U^1 < U^2 < \dots < U^{m_1}$, then the set of courses to be offered can be partitioned into subsets D_k such that $D_k = \{j/\text{forecasted enrollment in course } j \leq U^k\}$. From this definition it can be noted that if $j \in D_k \Rightarrow j \in D_{k+i}$ $i=1,2,\dots,m_1-k$. Also, it can be noted that if $j \in D_k$, then the number of classrooms available in which course j could be offered at time i is given by $M_i^k = \sum_{L=k}^{m_1} N_i^L$. From the previous definitions, the constraints indicating the limitation on available classroom space may be expressed in the form:

$$\sum_{j \in D_k} x_{ij} \leq M_i^k \quad i=1,2,\dots,m; k=1,2,\dots,m_1$$

Category three expresses the constraints representing faculty conflicts. Faculty conflicts represented in this category are those which would occur if more than one course taught by the same faculty member was scheduled to be taught at the same time. Defining

$W_k = \{j/\text{course } j \text{ is taught by faculty member } k\}$ with $k=1,2,\dots,m_2$ where m_2 is the number of faculty members, then the constraints expressing faculty conflicts may be written as:

$$\sum_{j \in W_k} x_{ij} \leq 1 \quad i=1,2,\dots,m; k=1,2,\dots,m_2$$

Category four contains all constraints indicating conflicts that would be created for students if certain courses were to be scheduled at the same time. The most common type of conflicts that arise are those which occur when courses which must be taken by groups of students of the same type, such as juniors, seniors, etc., are taught at the same time. Any other constraints which serve to indicate that for one reason or another certain courses should not be offered at the same time may be included in category four. Assuming that m_3 sets of courses have been identified such that the courses appearing in any one set should not be taught simultaneously, the constraints in category four may be written in the form:

$$\sum_{j \in L_k} x_{ij} \leq 1 \quad i=1,2,\dots,m; k=1,2,\dots,m_3$$

where $L_k =$ index set of courses which may not be taught simultaneously. The model developed in the previous pages of this section can be formulated as follows:

$$\text{minimize } \sum_{j=1}^n \sum_{i=1}^m c_{ij} x_{ij}$$

subject to:

$$\begin{aligned}
 (1) \quad & \sum_{i=1}^m x_{ij} = 1 && j=1,n \\
 (2) \quad & \sum_{j \in D_k} x_{ij} \leq m_i^k && i=1,m; k=1,m_1 \\
 (3) \quad & \sum_{j \in W_k} x_{ij} \leq 1 && i=1,m; k=1,m_2 \\
 (4) \quad & \sum_{j \in L_k} x_{ij} \leq 1 && i=1,m; k=1,m_3 \\
 & x_{ij} = \text{binary} && i=1,m; j=1,n
 \end{aligned}$$

From the construction of the model, the sets over which the summations are carried out have some properties which will be of importance later on. These are:

1. $D_1 \subset D_2 \subset D_3 \subset \dots \subset D_{m_2}$
2. $W_\alpha \cap W_\beta = \phi \quad \forall \beta \neq \alpha.$
3. $\bigcup_{k=1}^{m_2} W_k = \text{set of all courses to be offered.}$
4. $D_{m_1} = \bigcup_{k=1}^{m_2} W_k$
5. $M_i^{m_1} = \text{total number of classrooms available at time } i.$

Figure 6 gives a graphical representation of the general structure of the constraint matrix of the course scheduling problem as formulated on the previous page. As can be noted, the constraints represented in

Category		$x_{11} \dots x_{m1}$	$x_{12} \dots x_{m2}$	$x_{13} \dots x_{m3}$	$x_{14} \dots x_{m4}$...	$x_{1n} \dots x_{mn}$
1	n rows, each row having exactly m entries. Any variable can only appear once in any row.	1 1 1	1 1 1	1 1 1	1 1 1	...	1 1 1
2	m_1 sets of m rows. From one set to another, the same entries appear, except that each successive set may have other entries.	I I I	I I	I I I	I	...	I
3	m_2 sets of m rows. Any entry may appear only once in any row.	I	I	I I			I
4	m_3 sets of m rows. Each set of rows has arbitrary entries.	I	I I	I I	I		I

Figure 6. Constraint Coefficient Matrix: Course-Time Scheduling Problem

category one contain m coefficients of one, each set of which represents a given course j offered at any time i . The constraints represented in category two can be grouped in sets of m constraints, each one corresponding to a given time period. As denoted by property one, each successive set of m constraints contains in it the entries of the previous set of m constraints. As such, the last set of m constraints forms a row of n identity matrices. The constraints represented in category three can also be grouped into sets of m constraints. Each constraint set corresponds to the courses being taught by a particular faculty member and, as indicated by property two, each variable appears in only one of the constraints of category three. In category four the constraints may again be grouped into sets of m constraints but there is no general characteristic which can be used to further describe the structure of the constraints belonging to this category.

It can be easily verified that the general course scheduling problem consists of mn variables and $m(m_1+m_2+m_3) + n$ constraints.

The mathematical aspects of the course scheduling problem formulation will be explored in the next section of this chapter.

Mathematical Aspects

If the constraints denoting that x_{ij} must be binary are neglected, then the course scheduling problem is a linear program. The constraint coefficient matrix of the linear program is composed entirely of zeros and ones. Since there are certain classes of problems which have constraint coefficient matrices composed entirely of zeros and ones and which have the property that any basic solution is integer (i.e.,

matrices with this characteristic are said to possess the unimodular property), the initial hope for the course scheduling problem was that it also possessed this property.

Several examples were solved by linear programming, and in each case the optimal solution was integer. Unfortunately, it was found that in some cases intermediate solutions were not integer. Since the objective function of the course scheduling problem is general in nature, the conclusion was that linear programming will not guarantee integer solutions in all cases. This conclusion was further corroborated by the fact that the structure of the problem is similar to the structure of the multi-commodity network flow problem as presented in Chapter I. The similarity can immediately be noted in that the constraints defined by (2), (3), and (4) in the formulation of Chapter I can be replaced by the constraints defined by (1) and (3) of the formulation presented in the previous section, and that the constraint corresponding to (5) of Chapter I can be replaced by the constraints (2) and (4) of the previous section. According to Hu [19], multi-commodity network flow problems do not have, in general, constraint coefficient matrices possessing the unimodular property.

In the remainder of this section it will be shown that the constraint set of the course scheduling problem can be partitioned into two sets, one possessing the unimodular property. Furthermore it will be shown that there are, in general, three different manners in which the constraints may be so partitioned. These will be denoted by the SIP, the MIP, and the LIP partitioning schemes, respectively.

The SIP Partitioning Scheme

The course scheduling problem may be rewritten in the form:

$$\text{minimize} \quad \sum_{j=1}^n \sum_{i=1}^m c_{ij} x_{ij}$$

subject to:

$$(1) \quad \sum_{i=1}^m x_{ij} = 1 \quad j=1, n$$

$$(2) \quad \sum_{j \in D_{m_1}} x_{ij} \leq M_i^{m_1} \quad i=1, m_1$$

$$(3) \quad \sum_{j \in D_k} x_{ij} \leq M_i^k \quad i=1, m; k=1, m_1-1$$

$$(4) \quad \sum_{j \in W_k} x_{ij} \leq 1 \quad i=1, m; k=1, m_2$$

$$(5) \quad \sum_{j \in L_k} x_{ij} \leq 1 \quad i=1, m; k=1, m_3$$

$$x_{ij} = \text{binary} \quad i=1, m; j=1, n$$

Recalling that:

$$D_{m_1} = \bigcup_{k=1}^{m_2} W_k = \{j/j=1, n\}$$

then it can be noted that constraint sets (1) and (2) are the

constraints corresponding to a standard transportation problem. It is a well-known fact that the constraint matrix of such a problem possesses the unimodular property. It is also well known that any transportation problem is equivalent to a problem of finding a maximum flow-minimum cost in a capacitated network. The type network involved is referred to as a bipartite one, since the set of nodes ($i=1,m; j=1,n$) may be divided into two subsets: $A = \{i/i=1,m\}$ $B = \{j/j=1,n\}$, such that all arcs leading from nodes in A go to nodes in B.

The SIP formulation has partitioned the constraints in a manner such that the resulting problem can easily be interpreted as being a transportation or network flow problem, subject to additional constraints. The basic reason for this partitioning is that very efficient procedures exist for the solution of transportation or network flow problems. Since the algorithm developed in the previous chapter utilizes the imbedded problem, in this case the network flow problem, to calculate the lower bounds on the subsets of the solution space, the computational efficiency attainable in solving the lower bounding problems will be largely responsible for the computational efficiency of the algorithm as a whole.

For latter use, it will be convenient to transform the bipartite network to circulation form. The general formulation of a network problem in circulation form is given by:

$$\text{minimize} \quad \sum_A a_{ij} f_{ij}$$

subject to:

$$f_{iN} - f_{Ni} = 0 \quad \forall i \in N$$

$$L_{ij} \leq f_{ij} \leq u_{ij} \quad \forall ij \in A$$

where:

N - set of nodes.

A - set of arcs.

a_{ij} - cost of a unit flow in arc ij .

L_{ij} - minimum permissible flow in arc ij .

u_{ij} - maximum permissible flow in arc ij .

f_{ij} - flow in arc ij .

The basic difference between the above representation and a bipartite network representation is that the above provides a network which is source and sink free while the latter does not. This property will be convenient for the computation of lower bounds as will be seen in later sections.

The general form of the constraint coefficient matrix of the imbedded problem under the present partitioning scheme is given in Figure 7. The corresponding general network representation associated with the imbedded problem is presented in Figure 8. Associated with this general representation, the quantities associated with the formulation of the network problem are defined as follows:

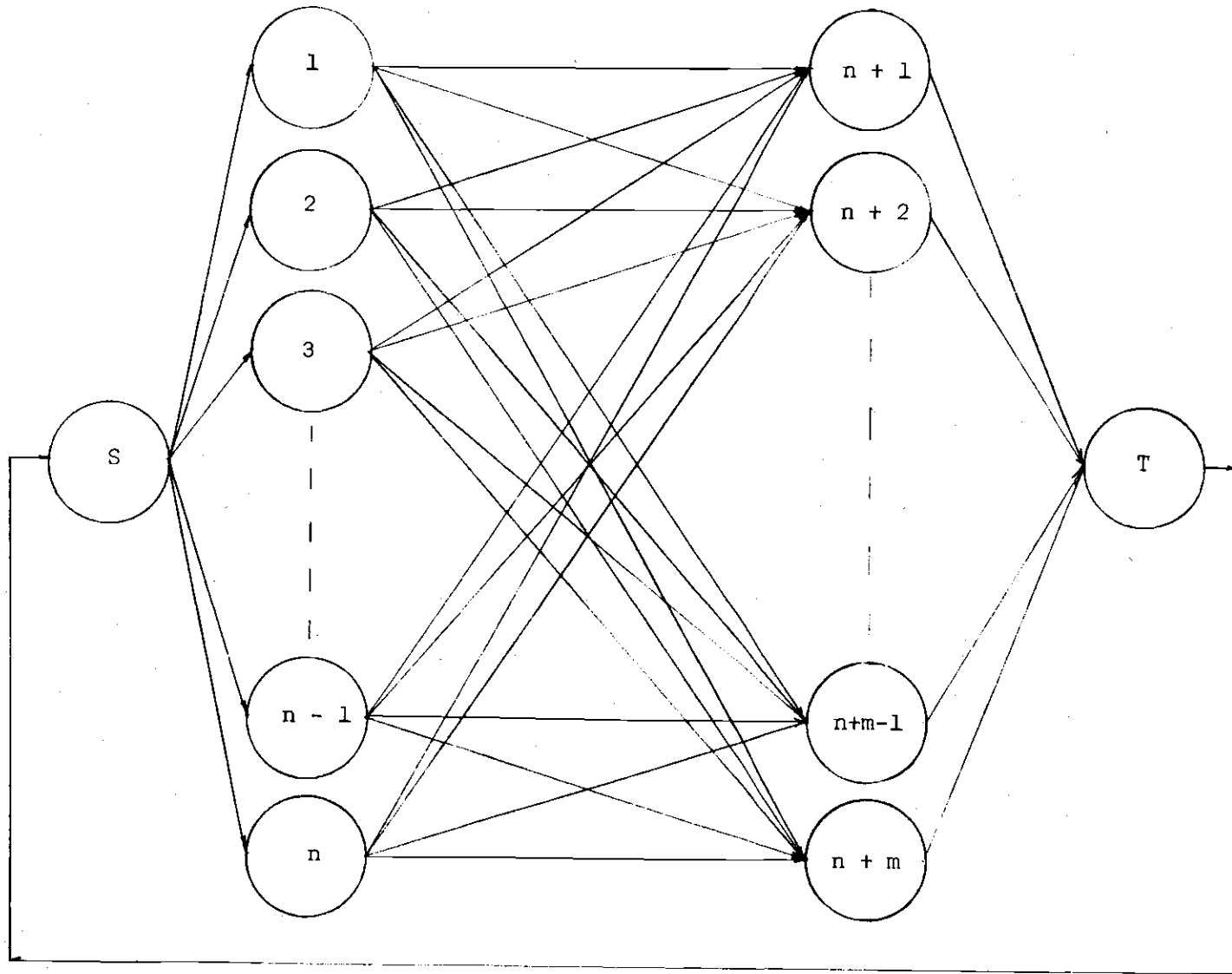


Figure 8. Network Representation of Imbedded Problem: SIP Partitioning Scheme

$$N = N_1 \cup N_2 \cup N_3 \cup N_4 \quad (1)$$

$$N_1 = \{S\} \quad (2)$$

$$N_2 = \{j/j=1,n\} \quad (3)$$

$$N_3 = \{i/i=n+1,r+m\} \quad (4)$$

$$N_4 = \{T\} \quad (5)$$

where:

N_2 = nodes corresponding to courses.

N_3 = nodes corresponding to times.

$$A = A_1 \cup A_2 \cup A_3 \cup A_4 \quad (6)$$

$$A_1 = \{ij/i=S, j=1,n\} \quad (7)$$

$$A_2 = \{ij/i=1,n; j=n+1,m+n\} \quad (8)$$

$$A_3 = \{ij/i=n+1,m+n; j=T\} \quad (9)$$

$$A_4 = \{ij/i=T; j=S\} \quad (10)$$

$$u_{ij} = L_{ij} = 1 \quad \forall ij \in A_1 \quad (11)$$

$$u_{ij} = 1, L_{ij} = 0 \quad \forall ij \in A_2 \quad (12)$$

$$u_{ij} = M_{i-n}^{m_1}, L_{ij} = 0 \quad \forall ij \in A_3 \quad (13)$$

$$u_{ij} = n, L_{ij} = 0 \quad ij \in A_4 \quad (14)$$

$$a_{ij} = 0 \quad \forall ij \in A_1 \quad (15)$$

$$a_{ij} = c_{j-n,i} \quad \forall ij \in A_2 \quad (16)$$

$$a_{ij} = 0 \quad \forall ij \in A_3 \quad (17)$$

$$a_{ij} = -\infty \quad \forall ij \in A_4 \quad (18)$$

Since in the original formulation of the course scheduling problem, x_{ij} assumes a value of one if course j is offered at time i , and zero otherwise, it follows that the equivalent statement in terms of network flows is that if $f_{ij} = 1$ for some $ij \in A_2$, then course i will be offered at time $j-n$. Consequently, the arc flows occurring in the set A_2 have a one to one correspondence with the decision variable x_{ij} of the original problem formulation. Definition (11) insures that any course can be taught at only one time, while definition (13) insures that the number of courses taught at any one time does not exceed the total number of classrooms available at that time. Definition (12) has been included in the network definition although at this point it does not add any

restrictions to the network. Later on this definition will allow for an expedient manner in which to introduce implicit constraints on the variables, deduced from the additional constraints, into the network problem.

For the purpose of determining the computational efficiency of the algorithm developed in Chapter II, the dimensionality of a problem will be defined in terms of the number of decision variables, x_{ij} , the size of the imbedded problem, and the number of additional constraints. The size of the imbedded problem will be defined in terms of the number of nodes and the number of arcs present in the circulation form of the network problem. For the SIP partitioning scheme the dimensionality of the problem is:

$$\begin{aligned}
 \text{number of variables} &= nm \\
 \text{number of nodes} &= m + n + a \\
 \text{number of arcs} &= n + mn + m + 1 \\
 \text{number of additional constraints} &= m(m_1 - 1 + m_2 + m_3)
 \end{aligned}$$

The MIP Partitioning Scheme

Consider the course scheduling problem as initially formulated, namely:

$$\text{minimize} \quad \sum_{j=1}^n \sum_{i=1}^m c_{ij} x_{ij}$$

subject to:

$$(1) \quad \sum_{i=1}^m x_{ij} = 1 \quad j=1, n$$

$$\sum_{j \in D_L - D_{L-1}} x_{ij} + s_i^L - s_i^{L-1} = N_i^L \quad i=1,m; L=2,m_1$$

Figure 9 graphically illustrates the structure of the coefficient matrix of the above constraints. Denoting the constraint coefficient matrix by A, and partitioning the rows of the matrix A into two disjoint sets T_1 and T_2 such that T_1 contains the first n rows, and T_2 the remaining rows, the unimodular property for the matrix can be claimed based on a theorem of Heller and Tompkins [18] which is quoted below:

Let A be an m by n matrix whose rows can be partitioned into two disjoint sets, T_1 and T_2 , such that A, T_1 and T_2 have the following properties:

Every entry in A is 0, +1, or -1; every column contains at most two non-zero entries; if a column of A contains two non-zero entries, and both have the same sign, then one is in T_1 and the other in T_2 ; if a column of A contains two non-zero entries, and they are of opposite sign, then both are in T_2 or both in T_1 ;

Then A has the unimodular property.

From Figure 9 it can be noted that the coefficient matrix satisfies all the conditions stated in the above theorem; hence the MIP partitioning scheme does indeed yield an imbedded constraint whose coefficient possesses the unimodular property.

The imbedded problem in the MIP partitioning scheme can also be represented as a maximum flow-minimum cost problem in a capacitated network. Correspondingly, as with the SIP partitioning scheme, the network problem can be put in circulation form. Its general representation is as shown in Figure 10. With this representation the quantities associated with the network formulation may be defined as follows:

Row Set		$x_{11} \dots x_{m1}$	$x_{12} \dots x_{m2}$	$x_{13} \dots x_{m3}$	$x_{14} \dots x_{m4}$...	$x_{1n} \dots x_{mn}$	S^1	S^2	$S^3 \dots S^l$
T_1		1 1...1	1 1...1	1 1...1	1 1...1		1 1...1			
T_2	Entries for all courses contained in D_1	I						I		
	Entries for all courses in $D_2 - D_1$		I					-I	I	
	Entries for all courses in $D_{m_1} - D_{m_1-1}$			I		...	I			-I I

Figure 9. Constraint Coefficient Matrix of Imbedded Problem: MIP Partitioning Scheme

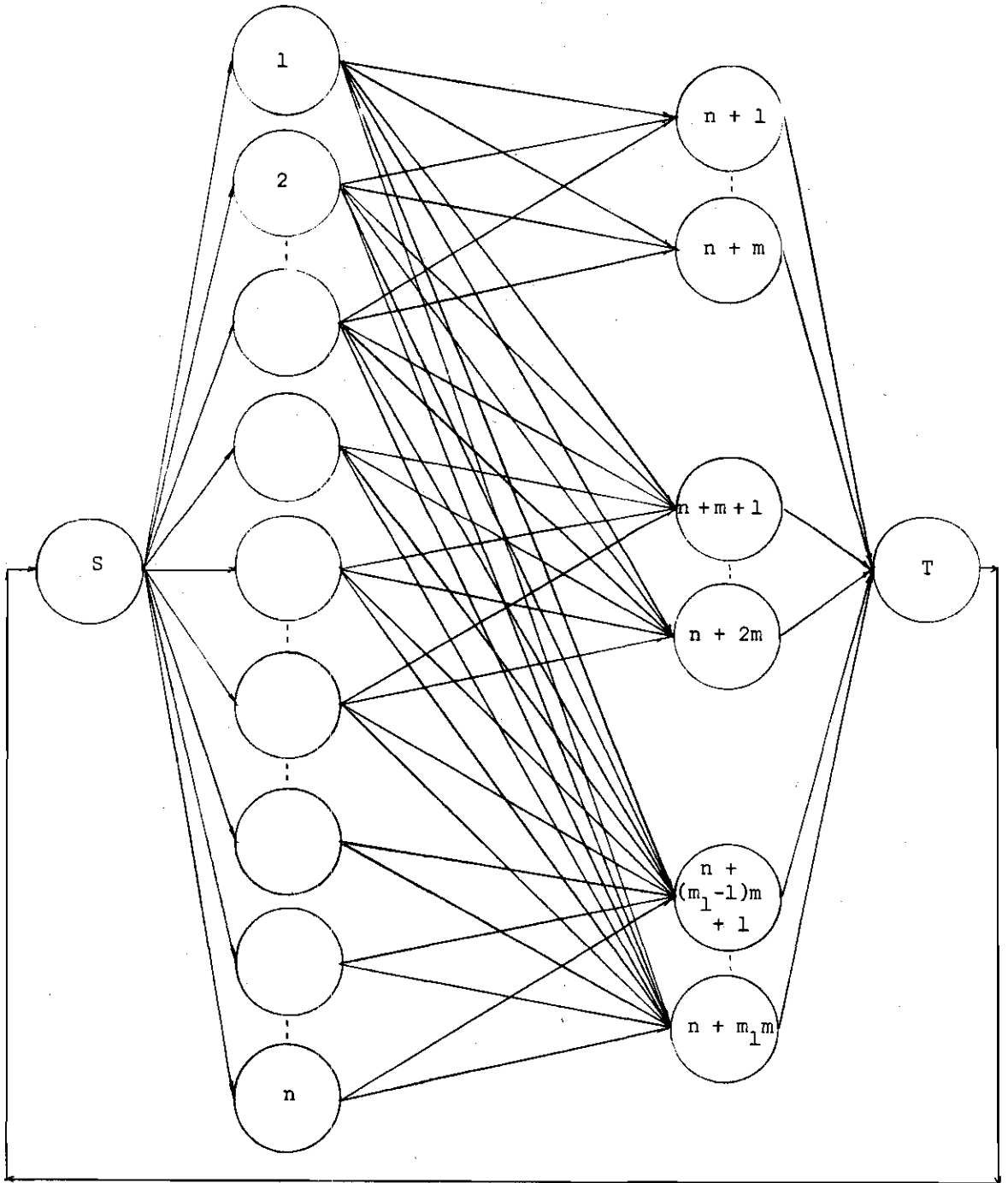


Figure 10. Network Representation of Imbedded Problem: MIP Partitioning Scheme

$$N = N_1 \cup N_2 \cup N_3 \cup N_4 \quad (1)$$

$$N_1 = \{S\} \quad (2)$$

$$N_2 = \{j/j=1,n\} \quad (3)$$

$$N_3 = \{i/i=n+1, \quad m_1+n\} \quad (4)$$

$$N_4 = \{T\} \quad (5)$$

where:

N_2 = nodes corresponding to courses.

N_3 = nodes corresponding to times.

$$A = A_1 \cup A_2 \cup A_3 \cup A_4 \quad (6)$$

$$A_1 = \{ij/i=S, j=1,n\} \quad (7)$$

$$A_2 = \{ij/i=1,n, j=n+(k-1)m+1, n+m_1 \text{ for } k \in D_k\} \quad (8)$$

$$A_3 = \{ij/i=n+1, n+m-m_1, j=T\} \quad (9)$$

$$A_4 = \{ij/i=T, j=S\} \quad (10)$$

$$u_{ij} = L_{ij} = 1 \quad \forall ij \in A_1 \quad (11)$$

$$u_{ij} = 1, L_{ij} = 0 \quad \forall ij \in A_2 \quad (12)$$

$$u_{ij} = N_{\beta}^{\alpha}, L_{ij} = 0 \quad \forall ij \in A_3 \quad (13)$$

where:

$$\alpha = \left\langle \frac{i-n+(m-1)}{m} \right\rangle$$

$\langle x \rangle =$ greatest integer $\leq x$

$$\beta = i - n - (\alpha-1)m$$

$$u_{ij} = n, L_{ij} = 0 \quad \forall ij \in A_4 \quad (14)$$

$$a_{ij} = 0 \quad \forall ij \in A_1 \quad (15)$$

$$a_{ij} = c_{\gamma i} \quad \forall ij \in A_2 \quad (16)$$

where:

$$\gamma = j - n - (\xi-1)m; \xi = \left\langle \frac{j-n+(m-1)}{m} \right\rangle$$

$$a_{ij} = 0 \quad \forall ij \in A_3 \quad (17)$$

$$a_{ij} = -\infty \quad \forall ij \in A_4 \quad (18)$$

In the above formulation of the imbedded problem network formulation, the correspondence between the arc flows $f_{\kappa\lambda}$ and the original decision variables x_{ij} of the course scheduling problem is somewhat different

than the SIP formulation. This stems from the fact that for each subset of courses $D_L - D_{L-1}$, a complete set of nodes corresponding to available times was introduced. As such, there are m_1 repetitions of the set of m nodes representing time slots. Since, by definition (11), there is exactly one unit of flow going to any node representing a course, it follows that the total flow leaving a course node must be one and, furthermore, that unit can flow to one and only one time node. The classroom availabilities are expressed as upper limits on the flows leaving the nodes representing time slots. The possibility of teaching courses contained in D_1 in classrooms with capacities greater than M_i^1 is considered by the arcs going to nodes representing time slots numbered greater than $n+m$. The same holds true for courses contained in any other subset except D_{m_1} . For courses in D_{m_1} , there are only $N_i^{m_1}$ classrooms available, which is denoted by only having arcs leaving any node corresponding to courses contained in D_{m_1} going to last m nodes representing time slots.

The dimensionality of a course scheduling problem using the MIP partitioning scheme is as follows:

$$\text{number of variables} = nm$$

$$\text{number of nodes} = n + mm_1 + a$$

$$\text{number of arcs} = n + m \left[\sum_{i=1}^{m_1} \left(\sum_{j \in D_i} 1 \right) (m_1 + 1 - i) \right] + m_1 m + 1$$

$$\text{number of additional constraints} = m(m_2 + m_3)$$

The LIP Partitioning Scheme

Consider the course scheduling problem formulation as follows:

$$\text{minimize} \quad \sum_{j=1}^n \sum_{i=1}^m c_{ij} x_{ij}$$

subject to:

$$(1) \quad \sum_{i=1}^m x_{ij} = 1 \quad j \in W_k; \quad k=1, m_2$$

$$(2) \quad \sum_{j \in W_k} x_{ij} \leq 1 \quad i=1, m; \quad k=1, m_2$$

$$(3) \quad \sum_{j \in D_k} x_{ij} \leq M_i^k \quad i=1, m; \quad k=1, m_1$$

$$(4) \quad \sum_{j \in L_k} x_{ij} \leq 1 \quad i=1, n; \quad k=1, m_3$$

$$x_{ij} = \text{binary} \quad i=1, m; \quad j=1, n$$

Recalling that $W_\alpha \cap W_\beta = \phi \quad \forall \alpha \neq \beta$, it is immediately apparent that the constraint sets defined by (1) and (2) define m_2 independent transportation problems and, consequently, m_2 independent network flow problems. As a result, the LIP partitioning scheme as presented above contains imbedded in its constraint set a subset of constraints whose coefficient matrix possesses the unimodular property.

For this partitioning scheme a decision must be made as to whether the imbedded problem should be considered as m_2 independent subproblems, or to join these into one larger problem. For the purpose of this thesis, the latter decision was taken.

The general representation of the network problem in circulation form is presented in Figure 11: the quantities associated with the network formulation may be defined as follows:

$$N = N_1 \cup N_2 \cup N_3 \cup N_4 \quad (1)$$

$$N_1 = \{S\} \quad (2)$$

$$N_2 = \{j/j=1,n\} \quad (3)$$

$$N_3 = \{i/i=n+1,n+mm_2\} \quad (4)$$

$$N_4 = \{T\} \quad (5)$$

where:

N_2 = nodes corresponding to courses.

N_3 = nodes corresponding to times.

$$A = A_1 \cup A_2 \cup A_3 \cup A_4 \quad (6)$$

$$A_1 = \{ij/i=S, j=1,n\} \quad (7)$$

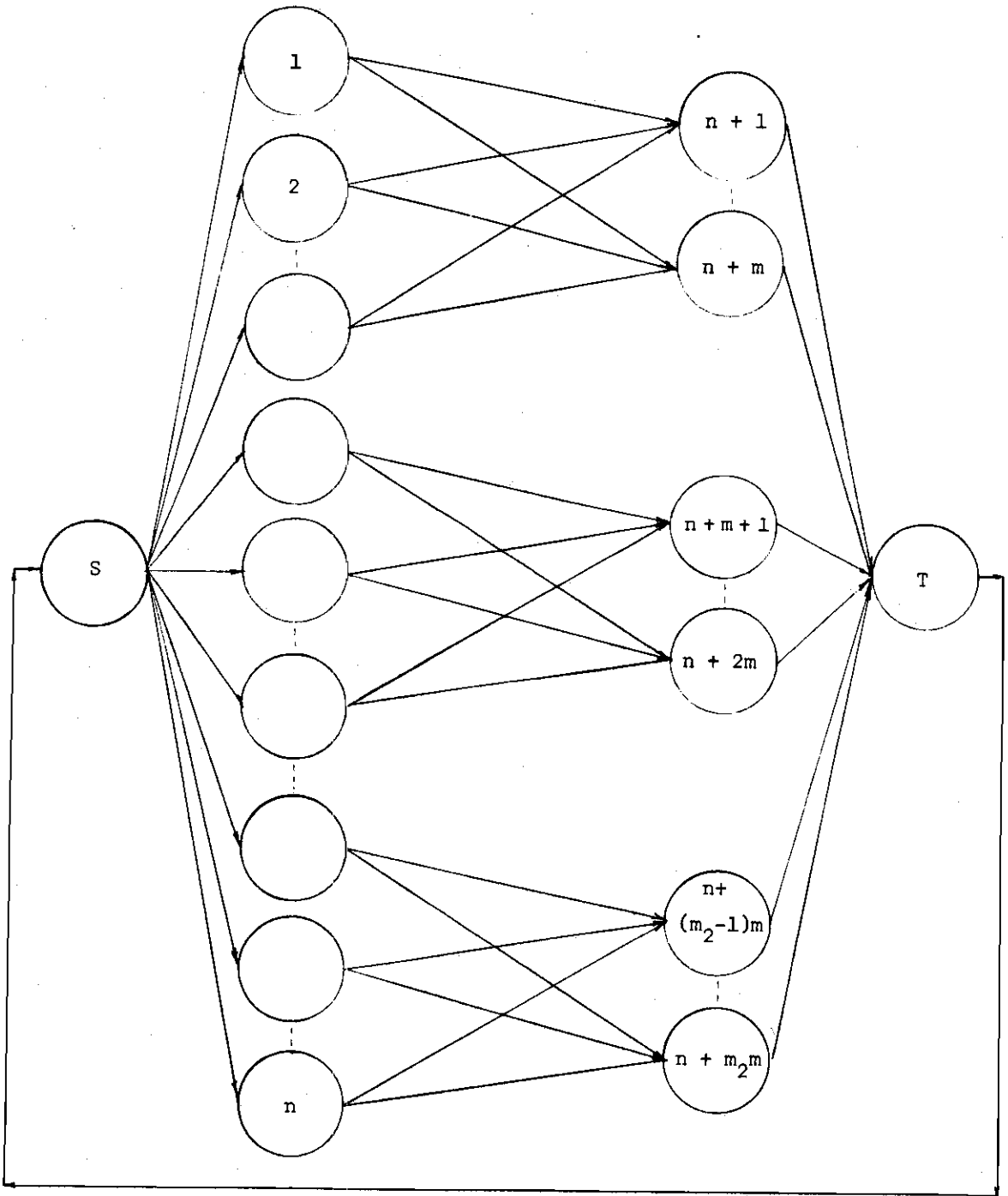


Figure 11. Network Representation of Imbedded Problem: LIP Partitioning Scheme

$$A_2 = \{ij/i \in W_k \text{ and } j = n + (k-1)m + 1, \quad (8)$$

$$(k-1)m + m + n \text{ and}$$

$$k = 1, m_2\}$$

$$A_3 = \{ij/i = n + 1, m_2 + n, j = T\} \quad (9)$$

$$A_4 = \{ij/i = T, j = S\} \quad (10)$$

$$u_{ij} = L_{ij} = 1 \quad \forall ij \in A_1 \quad (11)$$

$$u_{ij} = 1, L_{ij} = 0 \quad \forall ij \in A_2 \quad (12)$$

$$u_{ij} = 1, L_{ij} = 0 \quad \forall ij \in A_3 \quad (13)$$

$$u_{ij} = n, L_{ij} = 0 \quad \forall ij \in A_4 \quad (14)$$

$$a_{ij} = 0 \quad \forall ij \in A_1 \quad (15)$$

$$a_{ij} = c_{\alpha i} \quad \forall ij \in A_2 \quad (16)$$

where:

$$\alpha = J - (k-1)m - n \text{ for } k \ni i \in W_k.$$

$$a_{ij} = 0 \quad \forall ij \in A_3 \quad (17)$$

$$a_{ij} = -\infty \quad \forall ij \in A_4 \quad (18)$$

From definition (8) it can be noticed that for a specific k , representing a faculty member of the school, the set of arcs contained in A_2 go from every $i \in W_k$ to the nodes defined by $j = n+(k-1)m+1, n+(k-1)m+m$. This set of arcs defines a transportation problem where the sources are the m time slots, and the destinations the courses taught by faculty member k . The m_2 independent transportation problems have been merged into one by the introduction of the nodes S and T , and by adjoining the arcs contained in A_1, A_2 , and A_4 to the independent transportation problems.

Using the LIP partitioning scheme, the dimensionality of the class scheduling problem is:

$$\text{number of variables} = nm$$

$$\text{number of nodes} = n + m_2 m + 2$$

$$\text{number of arcs} = n + \sum_{k=1}^{m_2} \left[\sum_{j \in W_k} 1 \right] m + m m_2 + 1$$

$$\text{number of additional constraints} = m[m_1 + m_3]$$

Relevance of Partitioning Schemes

In the previous sections of this chapter three alternate partitioning schemes have been presented for the course scheduling problem. For each scheme, the partitioning of the constraint set was such that a set of constraints, the coefficient matrix of which possessed the unimodular property, were placed in one group and the remaining constraints in another. Further analysis of the constraints in the first group revealed that the problem defined by these imbedded constraints together

with the objective function could be expressed as an equivalent problem of finding the minimum cost-maximum flow in a capacitated network.

The dimensionality of a course scheduling problem was defined for each partitioning scheme in terms of the number of variables, x_{ij} , of the problem, the number of arcs and nodes in the network representation of the imbedded problem and the number of additional constraints or, in other words, the number of constraints that are not included in the imbedded problem formulation. Table 1 summarizes the dimensionality of the problem for each partitioning scheme.

The relationship between the SIP and the MIP partitioning schemes is quite clear. While the SIP formulation has a smaller imbedded problem in terms of nodes and arcs than the MIP formulation, the number of additional constraints is significantly larger than in the MIP formulation. The constraints forming the imbedded problem in the MIP formulation include the constraints of the imbedded problem in the SIP formulation. Consequently, the lower bounds obtained by using the MIP partitioning scheme are at least as "tight" as the lower bounds obtained if the SIP partitioning scheme was used. Generally speaking, it is desirable to use the procedure which gives the "tightest" lower bounds when utilizing an implicit enumeration approach. Nevertheless, it is important to recognize that the size of the imbedded problem of the MIP partitioning scheme is much larger than the imbedded problem of the SIP formulation and, consequently, each lower bounding operation takes more time for the larger imbedded problem than for the smaller. In the next chapter, computational results will be presented for several problems

Table 1. Dimensionality of Course Scheduling Problem for Different Partitioning Schemes

	SIP	MIP	LIP
Variables	mn	mn	mn
Nodes	$m + n + 2$	$n + mm_1 + 2$	$n + m_2m + 2$
Arcs	$n + mn + m + 1$	$n + m_1m + 1 + \left[\sum_{i=1}^{m_1} \left(\sum_{j \in D_i} 1 \right) (m_1 + 1 - i) \right] m$	$n + \sum_{k=1}^{m_2} \left[\sum_{j \in W_k} 1 \right] m + mm_2 + 1$
Additional Constraints	$m(m_1 - 1 + m_2 + m_3)$	$m(m_2 + m_3)$	$m(m_1 + m_3)$

solved using both the MIP and SIP partitioning schemes. These results are intended to provide an understanding of the behavior of the algorithm for the partitioning schemes, and give an indication as to which partitioning scheme is more desirable.

The LIP partitioning scheme contains different constraints in the imbedded problem than either the SIP or MIP formulation. Since the dimensionality of the imbedded problem using the LIP partitioning scheme is dependent on the subsets W_k and the number of faculty members (m_2) available to teach, it is difficult to deduce general statements about the relation between the LIP or the MIP and SIP formulations in terms of problem size, "tightness" of lower bound, or other measures of comparison.

In summary, three distinct manners of partitioning the course scheduling problem have been presented in detail. In the next chapter the computational effects of using these to solve specific problems will be presented. The specific algorithms used are derived from the general algorithm developed in Chapter II. Furthermore, various possible means of synthesizing two or more of the partitioning schemes will be presented together with a discussion of the computational advantages which can be obtained from such synthesis.

CHAPTER IV

APPLICATION-COMPUTATIONAL ASPECTS OF
THE COURSE SCHEDULING PROBLEM

This chapter will present the concepts and methodologies employed in developing a computational algorithm for the course scheduling problem as developed in the previous chapter. In addition, the computational results obtained by using the algorithm to solve various example problems will be presented.

The central concepts about which the algorithm is constructed are those presented in Chapter II. From a computational standpoint, these concepts are used to develop the Branch Help Table (BHT). The BHT is a dynamic representation of all the relevant information that can be extracted from the problem at some stage of computation.

The BHT contains the information presented in Figure 2 of Chapter II in tabular form. It is dynamic in the sense that at any point of the computation it contains only pertinent information and all non-pertinent information is discarded. Furthermore, at the beginning of the computation the table will be empty, and the algorithm terminates whenever the table becomes empty again.

In order to properly describe the BHT, it is first necessary to describe the manner in which the information describing the particular problem is organized for computational purposes. From data input, the information is organized for the network representation of the imbedded

problem and for the additional constraints. The next section of this chapter will describe the actual data organization.

Data Organization

The data input necessary in order to execute the course scheduling program must be in a specific sequence. The first elements of this sequence will consist of information obtained or generated from the Fcast and Profassign subsystems. With respect to the terminology developed in the previous chapter, the first elements of the data input sequence describe the subsets W_K for all K . In addition, it associates the name of the faculty member K with each subset of courses W_K .

The next elements of the data input sequence contain information, derived from the ISEMIS data base, describing the subsets $D_L - D_{L-1}$ for $L = 1, m_1$. Included with the information describing each of the subsets $D_L - D_{L-1}$ will be the associated N_i^K .

Next in the input sequence will be the descriptions of the subsets L_K , followed by a listing of the available time slots and of the costs associated with each.

The data input sequence as described above is presented in a general form in Table 2.

The above data input sequence will be utilized regardless of whether the SIP, MIP, or LIP partitioning schemes are employed to solve the course scheduling problem. Although the general concepts for the storage of the additional constraint and imbedded problem formulation will be retained for each partitioning scheme, the actual information stored will naturally vary depending on the partitioning scheme.

Table 2. Data Input Sequence Description

Sequence 1: Describes W_K 's	$\left\{ \begin{array}{l} \text{Name of Faculty Member 1} \\ \text{All } j \in W_1 \} \text{ courses taught by faculty member 1} \\ \\ \text{Name of Faculty Member } m_2 \\ \text{All } j \in W_{m_2} \} \text{ courses taught by faculty member } m_2 \end{array} \right.$
Sequence 2: Describes D_K 's	$\left. \begin{array}{l} \left. \begin{array}{l} m_1 \\ N_i^1, \\ \\ \text{All } j \in D_{m_1 - D_{m-1}} \end{array} \right\} \text{ number of classrooms available} \\ \\ \left. \begin{array}{l} \vdots \\ \\ N_i^1 \\ \text{All } j \in D_1 \end{array} \right\} \text{ courses in category } m_1 \end{array} \right\}$

Sequence 3: Describes L_K 's	$\left\{ \begin{array}{l} 1, \\ \text{All } j \in L_1 \} \text{ right-hand side} \\ \\ \left. \begin{array}{l} \vdots \\ \\ 1, \\ \text{All } j \in L_{m_3} \end{array} \right\} \text{ courses contained in category 1} \\ \\ \left. \begin{array}{l} \vdots \\ \\ 1, \\ \text{All } j \in L_{m_3} \end{array} \right\} \text{ right-hand side} \\ \\ \left. \begin{array}{l} \vdots \\ \\ 1, \\ \text{All } j \in L_{m_3} \end{array} \right\} \text{ courses contained in category } m_3 \end{array} \right.$
Time and Cost Info.	$\left\{ \text{time slot descriptor, cost} \right\}$ for all time slots

The information describing the additional constraints is subdivided into the right-hand side information and the constraint coefficient matrix. All the information describing the constraint coefficient matrix can be very compactly stored in a one-dimensional array in binary form. Furthermore, it is only necessary to store information about the first constraint of each constraint set. For example, one constraint set corresponding to the faculty conflicts is normally written as:

$$\sum_{j \in W_K} X_{ij} \leq 1 \quad i=1,m; \text{ some specific } K$$

All information necessary to generate the above constraints is contained in the single constraint:

$$\sum_{j \in W_K} X_{1j} \leq 1 \quad \text{for some specified } K$$

because, if the constraint matrix is written such that the first m columns correspond to X_{i1} $i=1,m$, the next m corresponds to X_{i2} $i=1,m$, etc., then the above single constraint indicates the first element of the set of identity matrices in the constraint coefficient matrix which describes those constraints.

Based on the above, all the information describing the additional constraint's coefficient matrix is contained in a single array of the form:

Entry, Row, Column.

Entry refers to the array element being described, Row refers to the particular constraint set being described, and Column refers to the particular set of variables being described. In all cases, the value of Row represents a given K. For example, if the SIP partitioning scheme is being utilized, then the constraints:

$$\sum_{i=1}^m X_{ij} = 1 \quad j=1,n$$

$$\sum_{j \in D_{m_1}} X_{ij} \leq M_i^{m_1} \quad i=1,m$$

will be included in the imbedded problem. The additional constraints will be:

$$\sum_{j \in D_K} X_{ij} \leq M_i^K \quad i=1,m, k=1, m_1-1$$

$$\sum_{j \in W_K} X_{ij} \leq 1 \quad i=1,m; k=1,m_2$$

$$\sum_{j \in L_K} X_{ij} \leq 1 \quad i=1,m; K=1,m_3$$

Supposing that it was desirable to keep the above constraints in the same order when constructing the additional constraint coefficient matrix, Row could assume a value from one to $(m_1-1) + m_2 + m_3$. Each value of Row will represent the m constraints associated with the actual constraint set represented. Each value of Column will represent m variables, namely X_{iRow} , $i=1,m$. Consequently, with respect to the

constraint coefficient matrix, a given value for Row and Column represents an m by m identity matrix in the additional constraint coefficient matrix.

Information concerning the right-hand sides of the additional constraints is stored in a one-dimensional array with as many entries as additional constraints.

The information defining the network representation is described below:

NODES: the total number of nodes in the network.

ARCS: the total number of arcs in the network.

STARTARC: beginning node of a particular arc.

ENDARC: ending node of a particular arc.

ARCCOST: cost of having one unit of flow in a particular arc.

MAXCAP: maximum permissible flow in a particular arc.

MINCAP: minimum permissible flow in a particular arc.

PI: value of dual variable associated with a particular arc.

FLOW: actual flow in a particular arc.

For a given partitioning scheme, the number of nodes and arcs can be computed by the relationships presented on Table 1 of Chapter III. The nodes are numbered sequentially starting with the first node representing a course, and ending with the node S . Arcs are numbered sequentially starting with the arc leaving node S to node one and ending with the arc leaving node T going to node S . Although the order in which the courses appear as nodes will vary depending on the partitioning scheme utilized, the set of arcs appearing between the course

and the time nodes will always have a correspondence with the decision variables X_{ij} .

The Branch Help Table

The BHT is intended to provide all the pertinent information about a problem at a given stage of computation. The computational aspects of the course scheduling solution algorithm are directed toward obtaining the necessary information so that it may be inserted in the BHT. The BHT is a single array of variable length containing, in binary form, the following information:

- a. Level of computation.
- b. Arc utilized at that level of computation to determine further partitioning of solution space S_1 .
- c. Time represented by arc.
- d. Course represented by arc.
- e. Lower bound on solution space subset with flow in arc set a value zero, and all flows in arcs represented by previous levels fixed at a given value.
- f. Lower bound on solution space subset with flow in arc set at one, all flows in arcs represented by previous levels of computation set at given values, and all implicit arc flows set at zero.
- g. Indicator indicating feasibility of flows, obtained in lower bounding operation resulting in e, to be feasible with respect to additional constraints or not.
- h. Same as above, except with reference to f.

i. Indicator showing at what value the flow in the arc should be fixed.

As can be noted, all the entries contained in the BHT have a direct interpretation with the algorithm presented in Chapter II. Furthermore, the information contained in the BHT is the same as the information presented in a general form on Figure 2 of Chapter II. Namely, the levels contained in the BHT are the same, the subsets Γ^i and $\bar{\Gamma}^i$ are defined by entries b, c, d, and I of the BHT. The BHT entries described by e and f represent the g^i and \bar{g}^i of Figure 2. The BHT entries described by g and h serve to indicate whether it is worthwhile to further explore a given branch in the search for optimality.

The above has been a brief description of the BHT. The next sections will describe in more detail the computational aspects of obtaining the necessary information for the BHT.

Construction of the BHT

The BHT is constructed starting with the level zero. The information contained in this level is the result of solving the imbedded problem without regard to the additional constraints. Recalling that the imbedded problems associated with the particular application being discussed can always be represented as network flow problems, the particular technique utilized to solve the network problem is the out of kilter algorithm developed by Ford and Fulkerson [14].

The out of kilter algorithm was selected as a tool to solve the network flow problems because it is extremely general in procedure. It assumes lower bounds as well as capacities on each arc flow, the cost

coefficients can be arbitrary in sign, and the method can be initiated with any circulation, feasible or not and any set of dual variables, or node numbers.

The above presented properties also make the out of kilter algorithm an ideal tool to solve the lower bounding problems. The freedom to begin with any circulation and any set of node numbers implies that from one level of computation to the next, the previous optimal flows may be used as a starting point to calculate the new optimal flows.

The property that the out of kilter algorithm assumes lower bounds as well as capacities for each arc flow is another reason why the out of kilter algorithm is ideally suited for the algorithm. Consider, for example, that variable X_{ij} is to be fixed at a value of zero. This restriction can directly be specified in the network problem by setting the flow capacities of the arcs representing X_{ij} at zero.

Before the imbedded problem is solved the first time using the out of kilter algorithm, the initial flow values for the arcs and the initial dual variables are assigned a value of zero. The solution to the imbedded problem consists of the optimal flows and dual variables. Conceptually, this process is represented on Figure 12.

Before proceeding with the construction of the BHT, it is necessary to determine whether the solution to the out of kilter algorithm is the optimal solution to the problem. Before this can be done, it is necessary to relate the Flow $[I]^*$ and the X_{ij} . The actual conversion from flows to decision variables is dependent on the partitioning scheme utilized.

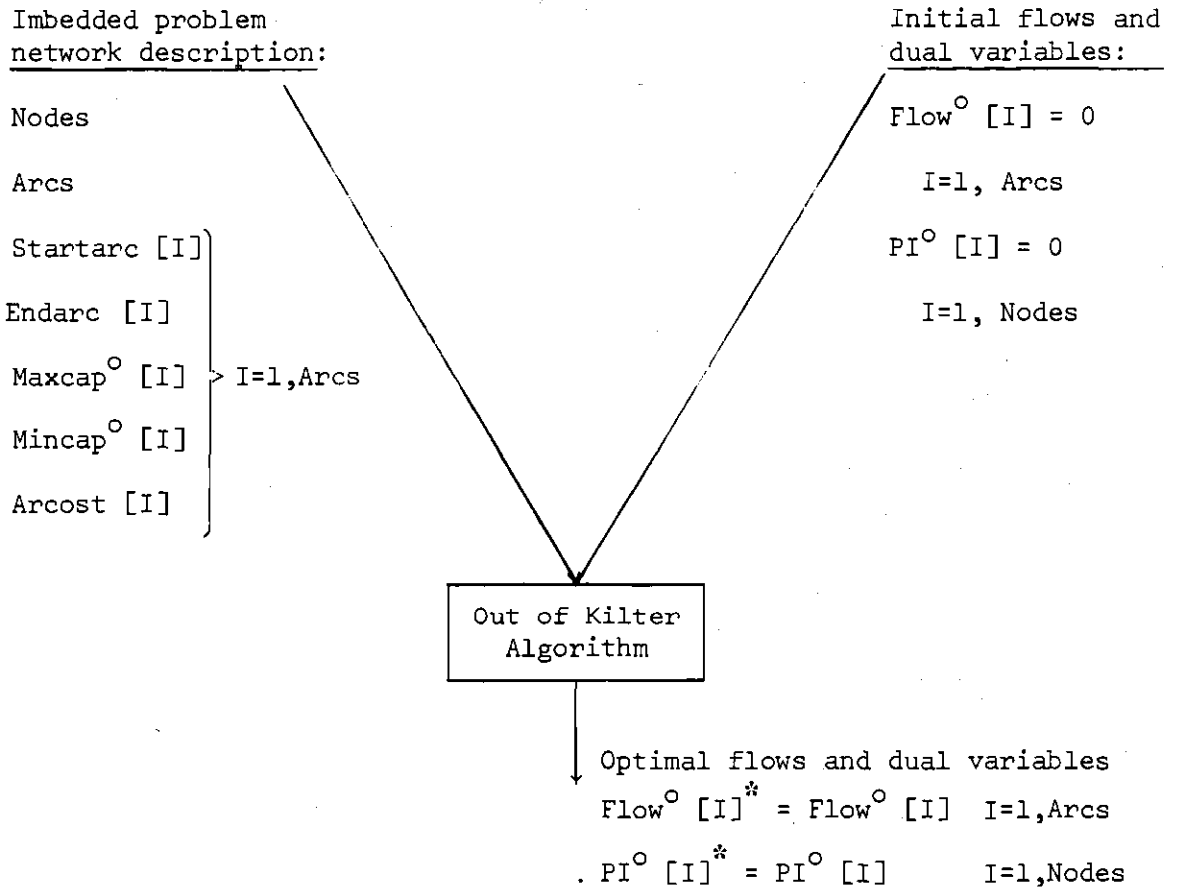


Figure 12. Computations Required at Level Zero of BHT.
 Upper Script Zero Implies Parameters
 Dependent on Level of Computation

From the construction of the network, the following correspondences can be noted:

For arc I:

$$\text{SIP: } \begin{cases} j = \text{Startarc [I]} & I \in A_2 \\ \text{no correspondence} & \text{otherwise} \\ i = \text{Endarc [I]} - n & I \in A_2 \\ \text{no correspondence} & \text{otherwise} \end{cases}$$

$$\begin{array}{l}
 \text{MIP} \\
 \text{and:} \\
 \text{LIP}
 \end{array}
 \left\{ \begin{array}{ll}
 j = \text{Startarc } [I] & I \in A_2 \\
 \text{no correspondence} & \text{otherwise} \\
 \\
 i = \text{Endarc } [I] - A_m - n & I \in A_2 \\
 \text{where:} & \\
 \\
 A = \text{greatest integer } \leq \left[\frac{\text{Endarc } [I] - (n+1)}{m} \right] & \\
 \text{no correspondence} & \text{otherwise}
 \end{array} \right.$$

Denoting the decision variables obtained from the Flow^o [I]^{*} by X_{ij}^{o*} , then the optimal solution has been obtained if $X^{o*} = \{X_{11}^{o*} X_{12}^{o*} \dots X_{mn}^{o*}\} \in S$; otherwise, the BHT is filled for other levels of computation.

After the results of the computation described above have been entered in the BHT, the subsequent levels are computed. The computational logic associated with applying the branching rule \bar{B} and the lower bounding procedures, as discussed in Chapter II, is illustrated in Figure 13. The computations are started by selecting arc $n+1$ as the initial arc to be fixed. From the results of level zero of the BHT, this arc will have either a flow of zero units or of one unit in it. If the flow in the arc is one, then the variable associated with the arc, denoted as $X_{ij(k)}$, is fixed at one, the implicitly fixed variables are found by applying the tests presented in Chapter II.

To find the lower bound on the subset of solutions to the imbedded problem with $X_{ij(k)} = 1$ and all implicitly fixed variables at a value of zero, the arc capacities and lower bounds are fixed in the network, and the out of kilter algorithm is used to find the new optimal flow through the network subject to the fixed variables.

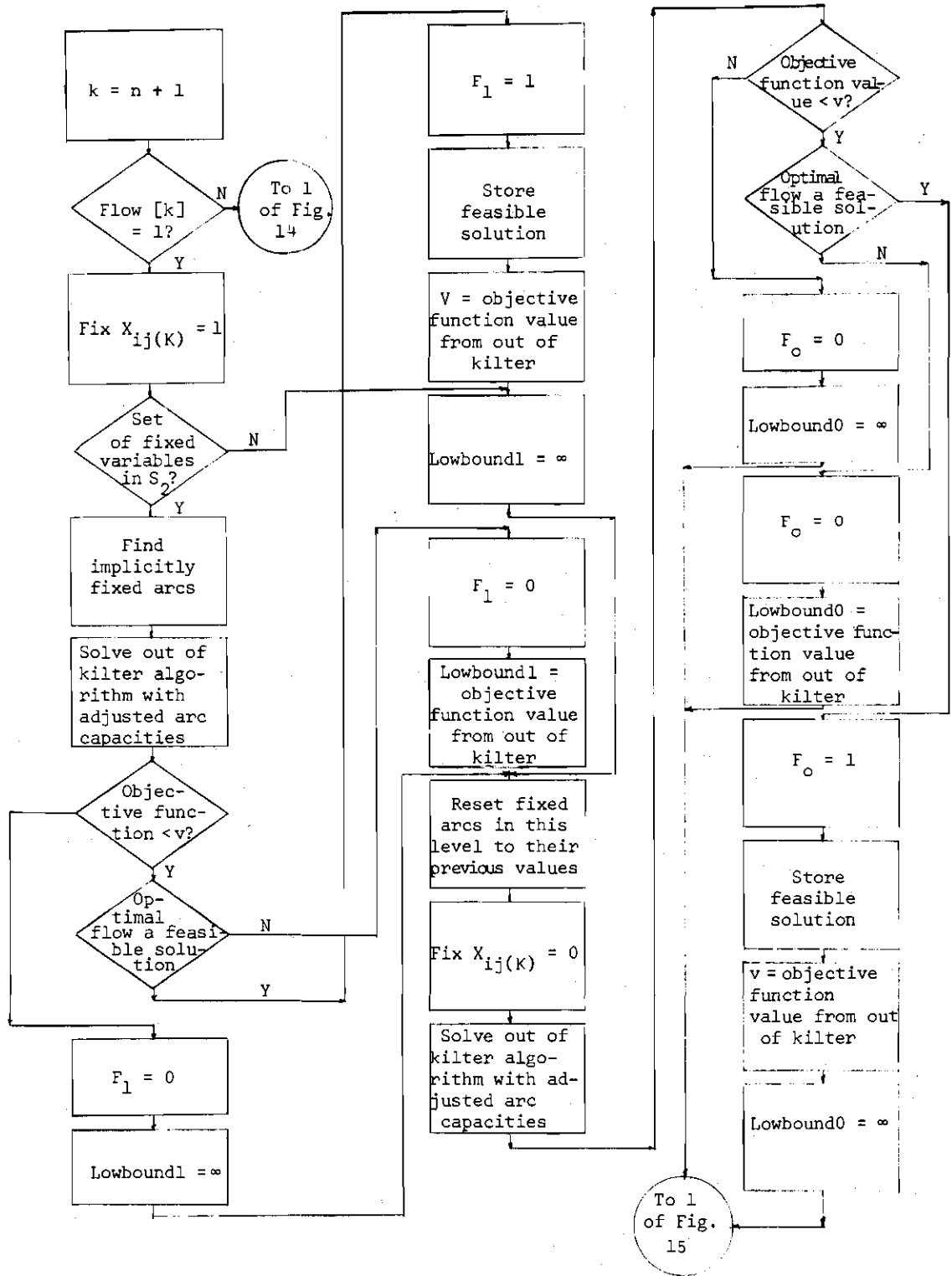


Figure 13. Application of Branching Rule and Bounding of Subsets - Part I

If the value of the objective function from the out of kilter algorithm is greater than the best feasible solution objective function value found so far, then the bound is set at ∞ . Otherwise the flows are converted to values of the X_{ij} 's and the additional constraints are checked for feasibility. If a feasible solution has been found, then it is stored and the bound on the subset is set at ∞ , otherwise the bound on the subset is given the value of the objective function from the out of kilter algorithm.

The bound on the subset generated by setting $X_{ij(K)}$ equal to zero is found in a similar manner after all arc capacities and low bounds have been reset to the values they had at the beginning of this level of computation. After the bounds of both subsets have been obtained, the decision must be made as to what subset to select for further branching. The logic associated with this selection will be discussed in a later paragraph.

If the original flow in arc K is zero, then the computations presented in Figure 14 must be performed. These are similar to those described on Figure 13, the basic difference being that the bound on the subset having $X_{ij(K)}$ equal to zero is already available from the previous level of computation. The bound on the subset having $X_{ij(K)}$ at one are computed in the same manner as exhibited in Figure 13.

Figure 15 exhibits the decision process required to select a subset for further partitioning. If both bounds are ∞ , then backtrack- ing is required, otherwise the subset with the smallest lower bound is selected for further partitioning. In order to initiate the next level

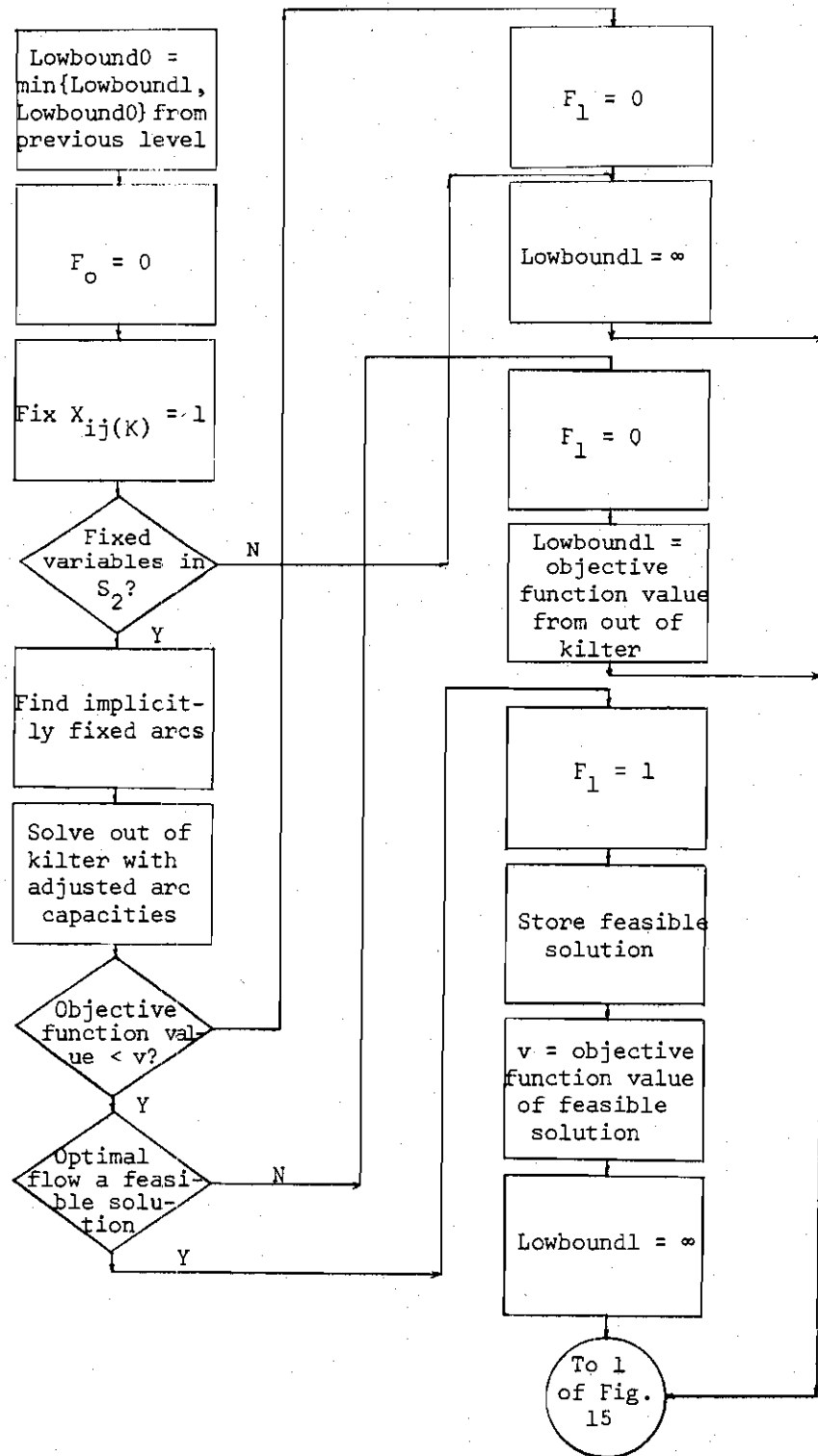


Figure 14. Application of Branching Rule and Bounding of Subsets - Part II

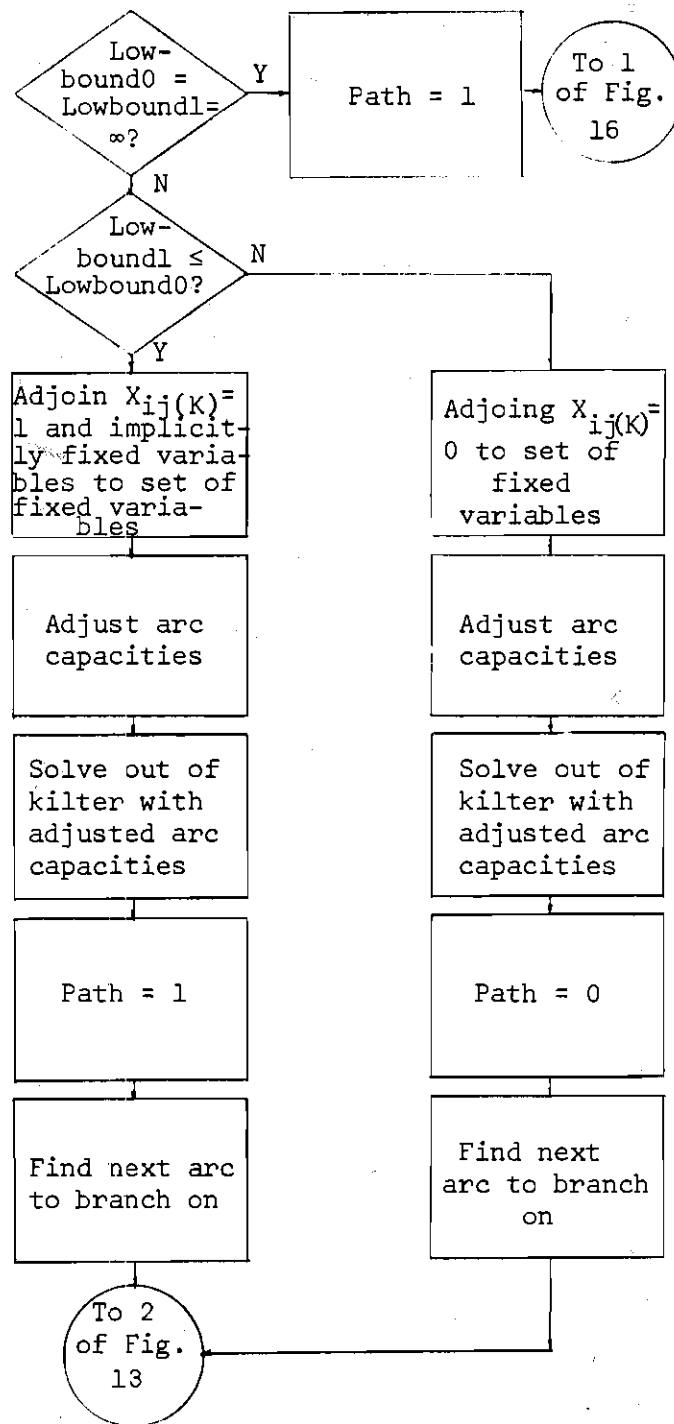


Figure 15. Decision Process for Choice of Subsets for Further Partitioning

of computation with the set of optimal flows associated with the given set of fixed variables, the out of kilter algorithm is used to optimize the flow. The computations described so far are then repeated, but the arc selected for partitioning is now the first arc for which the flow has not been fixed, either implicitly or explicitly.

If backtracking is necessary, the computations shown on Figure 16 are performed. The computations simply amount to examining the preceding level of computations in order to see if there is some active subset for which the lower bound is better than the objective function value of the best feasible solution found so far. If this is the case at some level, then the branch and bound process described in Figures 13 and 14 is applied again, otherwise the values of the bounds are made ∞ for that level, and the previous level is examined.

After the computations associated with a level are completed, the relationship of the results obtained to entries in the BHT may be observed in Table 3.

Table 3. Relation of BHT Entries to Computational Sequences

a	b	c	d	e	f	g	h	i
Level L	K	I(K)	J(K)	Lowbound0	Lowbound1	F_0	F_1	Path

This section of the thesis has presented the general computational sequences that must be performed in order to construct the BHT. For the various partitioning schemes presented in the previous chapter,

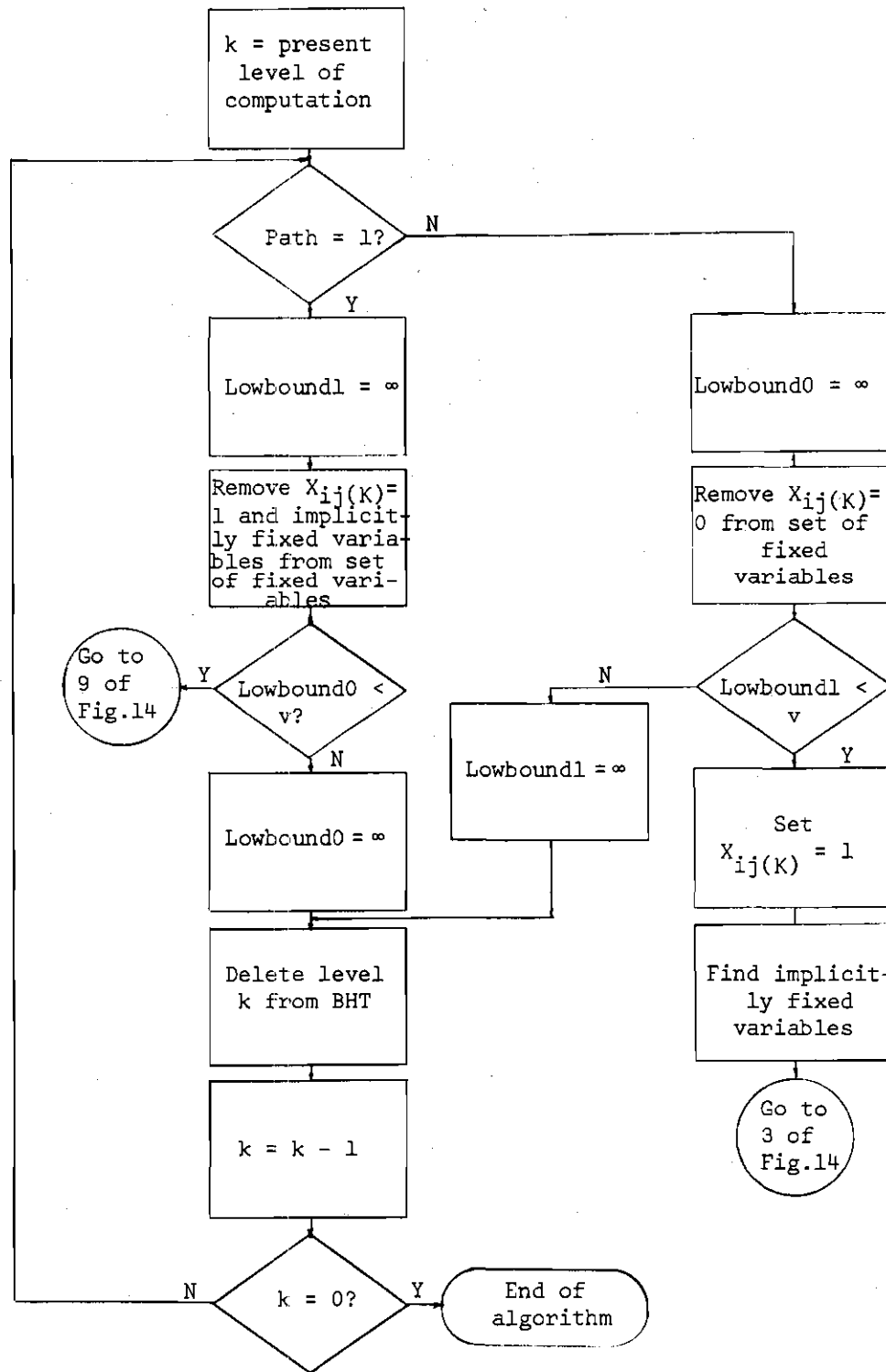


Figure 16. Backtrack Logic

some details are different, but will not be presented in detail. It is felt that with the information provided in this section the computational algorithms can be reconstructed without too much difficulty.

The next section of this chapter will present the computational results obtained with various computer programs written for the specific application discussed in this thesis.

Computational Analysis and Experience

Three separate computer programs were developed to obtain computational experience with the basic algorithm developed in Chapter II, specialized for the course scheduling problem presented in Chapter III. Although each computer program follows the computational sequence described in the previous section of this chapter, they are different in detail to incorporate the peculiarities of each scheme.

The programs have been written in Algol and all results have been obtained from a Burroughs B-5500 computer operating simultaneously in a batch and remote operations mode. Since the B-5500 is a multiprocessing machine, the times presented with the various solutions cannot be compared to those that can be obtained from a computer processing programs sequentially in that the processing times of the B-5500 are dependent on the particular load on the machine at that time, and also on the types of programs being processed simultaneously with the course scheduling program. The only general statement that can be made is that the presented times are slower than those that could be obtained from a similar machine processing programs sequentially.

The computational experience with three problems will be presented. Problem one is a relatively small problem which was utilized for testing purposes. Problem two is composed of data obtained from an actual scheduling situation and represents a fairly typical problem. Problem three is also composed of actual data, but it represents the largest problem that would normally be encountered. The following sections of this chapter will be devoted to describing each of these problems in greater detail and to presenting the computational results from the algorithm.

Problem 1

This problem was designed to test the computational algorithm of the course scheduling problem.

Table 4 summarizes the basic information about the problem together with the solution times obtained for each partitioning scheme.

Table 4. Summary Presentation of Problem One with Solution Times

Partitioning Scheme	No. of Variables	No. of Nodes	No. of Arcs	Number of Additional Constraints	Process Time	Input-Output Time	Total Time
SIP	30	14	43	35	0.547 min	0.253 min	0.80 min
MIP	30	18	62	30	4.18 min	0.106 min	4.28 min
LIP	30	24	52	25	0.460 min	0.241 min	0.70 min

Original integer programming formulation: 30 variables
46 constraints.

The running time information has been subdivided into processing, input-output, and total time. Since the Burroughs B-5500 computer is a multiprocessing machine utilizing dynamic overlay, the amount of time it spends allocating core storage is considered as input-output and is not present in serial processing computers. The total time is considered the sum of the processing and input-output time for the purposes of this thesis.

Problem 2

This problem has been taken from an actual set of course offerings and teaching assignments and could be considered as a fairly representative example of the majority of problems that would be encountered in the operation of the School of Industrial and Systems Engineering. Table 5 summarizes the relevant information about this problem together with the solution times obtained for each partitioning scheme.

Table 5. Summary Presentation of Problem Two with Solution Times

Partitioning Scheme	No. of Variables	No. of Nodes	No. of Arcs	Number of Additional Constraints	Process Time	Input-Output Time	Total Time
SIP	115	30	144	105	26.85 min	0.253 min	27.2 min
MIP	115	45	374	90	8.46 min	0.304 min	8.75 min
LIP	115	85	199	50	Optimality achieved, not verified > 60 min.		

Original integer programming formulation: 115 variables
135 constraints.

Problem 3

This problem is somewhat different from the previous in that in it laboratory sessions must be scheduled. Since many courses have associated with them more than one laboratory section, the number of variables in this problem is much larger. On the other hand, since laboratory sections normally have less students in them than lecture sections, the number of constraints indicating physical facility constraints are less than in the previous problem. Table 6 summarizes the relevant information of the problem together with the solution times obtained using each partitioning scheme.

Table 6. Summary Presentation of Problem Three with Solution Times

Partitioning Scheme	No. of Variables	No. of Nodes	No. of Arcs	Number of Additional Constraints	Process Time	Input-Output Time	Total Time
SIP	400	52	451	290	No optimality achieved > 60 min		
MIP	400	62	651	280	No optimality achieved > 60 min		
LIP	400	192	591	150	Optimality achieved--not verified > 60 min.		

Original integer programming formulation: 400 variables
340 constraints.

Up to this point the computational results were presented in a summary form. Since the algorithm finds a sequence of feasible solutions, each one having a better solution value than the previous, it is of interest to show the behavior of the algorithm for the various problems. From a user's standpoint, the interest arises due to the

trade-offs that are present between obtaining a feasible solution better than one found so far, and the additional cost in computational time involved with finding it. Tables 7 through 9 present the sequence of increasingly better feasible solutions found for each problem and partitioning scheme together with the computer time required to find each.

Table 7. SIP Partitioning Scheme: Detailed Solution Patterns and Times

Problem	Imbedded Problem $f(x)$	Feasible Solution Number	Feasible Solution $f(x)$	Process Time	Input-Output Time	Total Time
1	6	1	14	0.110 min	0.247 min	0.358 min
		2	12	0.233 min	0.249 min	0.482 min
		End	12	0.547 min	0.253 min	0.800 min
2	29	1	33	1.475 min	0.160 min	1.635 min
		2	31	2.222 min	0.165 min	2.490 min
		3	29	26.800 min	0.168 min	27.100 min
		End	29	26.850 min	0.170 min	27.200 min
3	70	1	76	10.800 min	0.633 min	11.48 min
		Termination Not Achieved				

In each of the tables, the first column gives reference to the particular problem, the second column presents the initial lower bound which is calculated by solving the imbedded problem without regard to the additional constraints. Columns three and four, respectively, present the feasible solutions found in their proper sequence and the

Table 8. MIP Partitioning Scheme: Detailed Solution Patterns and Times

Problem	Imbedded Problem f(x)	Feasible Solution Number	Feasible Solution f(x)	Process Time	Input-Output Time	Total Time
1	6	1	12	0.161 min	0.104 min	0.267 min
		End	12	4.180 min	0.165 min	4.280 min
2	29	1	33	2.970 min	0.294 min	3.260 min
		2	31	3.090 min	0.297 min	3.390 min
		3	29	8.450 min	0.302 min	8.740 min
		End	29	8.460 min	0.304 min	8.750 min
3	70	1	79	14.890 min	0.286 min	15.150 min
		2	77	14.950 min	0.289 min	15.300 min
		3	76	15.250 min	0.292 min	15.500 min
		Termination Not Achieved				> 60 min

objective function value associated with the respective feasible solutions. The remaining columns present a breakdown of the times required to achieve the solutions. As can be noted, problem three does not result in termination utilizing any one of the partitioning schemes individually. Except for the LIP partitioning scheme, all other problems terminate with optimality of the solutions verified. The LIP partitioning does not terminate for problems two or three.

Computational Synthesis

After studying the detailed solution patterns and times presented in the previous section, it became apparent that for the specific problems solved, considerable computational advantages could have been

Table 9. LIP Partitioning Scheme: Detailed Solution Patterns and Times

Problem	Imbedded Problem f(x)	Feasible Solution Number	Feasible Solution f(x)	Process Time	Input-Output Time	Total Time
1	6	1	12	0.1025 min	0.2400 min	0.330 min
		End	12	0.460 min	0.2410 min	0.700 min
2	23	1	35	1.700 min	0.123 min	1.825 min
		2	31	1.845 min	0.125 min	1.973 min
		3	29	3.90 min	0.129 min	4.050 min
		Termination Not Achieved				
3	53	1	72	14.300 min	0.393 min	14.700 min
		2	70	14.450 min	0.398 min	14.800 min
		Termination Not Achieved				

obtained by the synthesis of two or more of the partitioning schemes. The following paragraphs of this section will describe a specific synthesis which would have resulted in considerable computational improvements for the problems that were solved, and then to describe some other syntheses which could improve computational results for other problems.

Considering the results presented in Tables 8 and 9, it becomes apparent that the LIP scheme could have terminated with an optimal solution for problems two and three if some of the information obtained from the MIP scheme was made available a priori. Namely, given the imbedded problem solution obtained with the MIP scheme, the feasible solutions obtained through the use of the LIP scheme could have been compared with it, and as soon as a feasible solution was obtained with

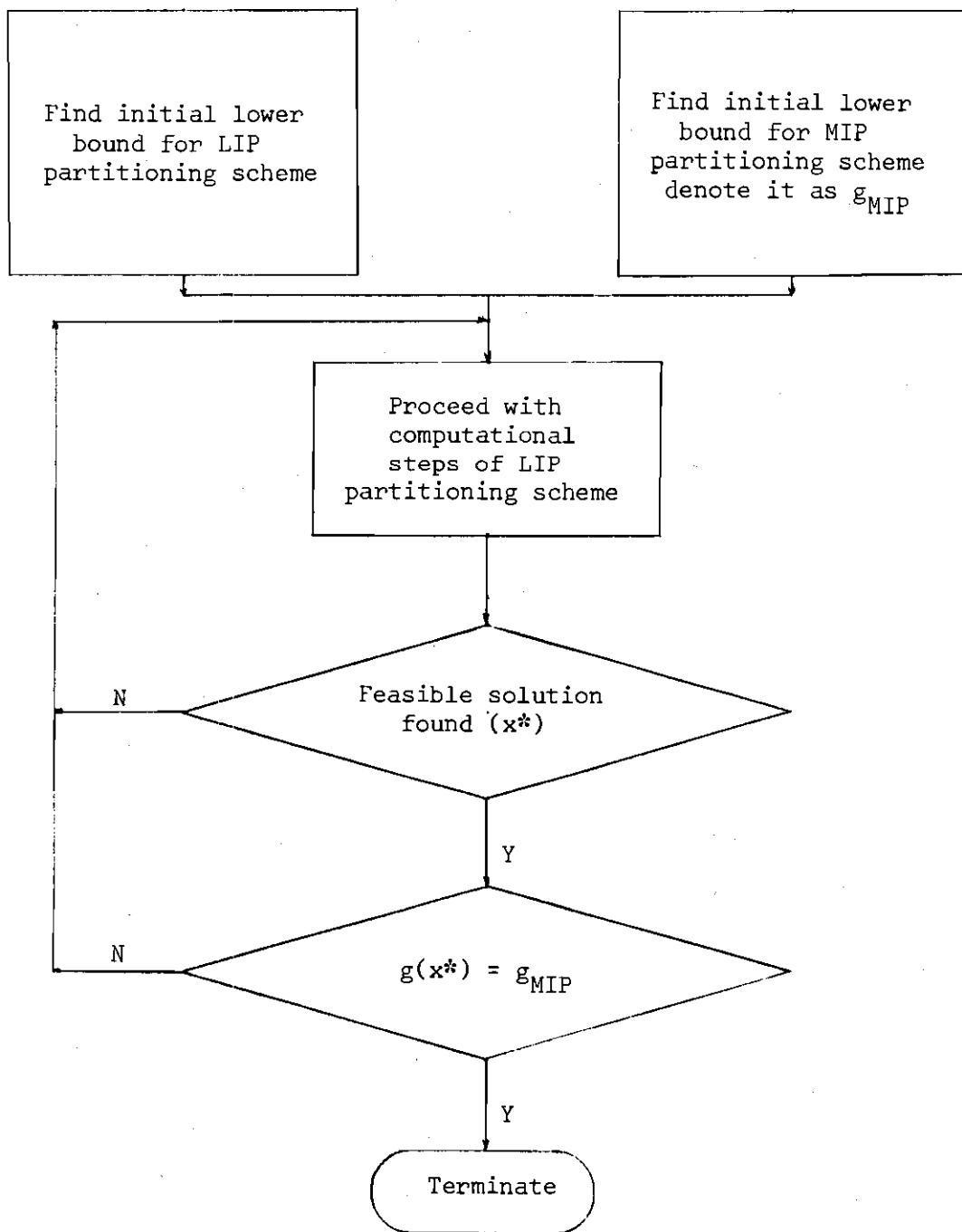


Figure 17. Computational Synthesis Scheme: LIP-MIP

an objective function value equal to the objective function value of the optimal solution to the MIP imbedded problem, the computations terminated. The additional computational times involved with such a scheme is marginal. The basic concepts of this particular synthesis are summarized in Figure 17, and the results obtained are exhibited in Table 10.

Table 10. Summary Results: Synthesis of LIP and MIP Partitioning Schemes

Problem	Variables	Constraints	Time to Optimal LIP	Additional Time MIP	Total Solution Time
1	30	46	0.70 min	0.026 min	0.703 min
2	115	133	4.05 min	0.186 min	4.236 min
3	400	340	14.80 min	0.472 min	15.272 min

The above presented synthesis was developed under the consideration that the initial lower bound obtained through the LIP partitioning scheme was lower than the lower bounds obtained with the MIP scheme. Furthermore, the imbedded problem of the LIP partitioning scheme is composed of constraints which have as their right-hand sides a value of one. Since there are m_2 of such constraints embodied in the imbedded problem, the possibility of obtaining a feasible solution when solving some intermediate lower bound problem may be better than when these "tight" constraints are considered as additional constraint in the manner of the SIP or MIP schemes.

The computational success of the above synthesis for the problems considered was primarily due to the fact that the optimal solution to the problems turned out to be an alternate optima to the imbedded problem of the MIP partitioning scheme. This cannot be guaranteed in all cases, as can be noted by the results of problem one.

Whenever the above synthesis is applied to problems for which the optimal solution is not an alternate optima of the MIP imbedded problem, then the results obtained will be identical to those obtained if just the LIP scheme was used; consequently, a more general synthesis procedure will be suggested. Before doing so it should be noted that the LIP scheme failed to terminate as a direct consequence of the "loose" bounds that were obtained. This can be verified by considering that the optimal solution was obtained after more than 150 levels of computations were performed. In order to verify optimality, the algorithm had to backtrack to each active node and branch on it until the bounds obtained were greater or equal to the optimal solution. Since the bounds calculated were "loose," the amount of branching required to obtain a high enough value of the lower bound was considerably more than with the other partitioning schemes and, naturally, more time consuming.

The concept utilized to develop the general synthesis consists of branching according to the LIP scheme, making a decision as to what subset to branch on next as in the LIP scheme, but then to utilize the SIP scheme to assign a lower bound to the subset which was not selected for further branching. Whenever a feasible solution is obtained as a

result of solving the bounding problems with the LIP or SIP scheme, the solution value is checked against the optimal solution value of the MIP imbedded problem.

The above concepts can be used whenever the initial bounds obtained by the LIP, SIP, and MIP scheme are ordered. In other words, when $\text{lowbound (LIP)} < \text{lowbound (SIP)} < \text{lowbound (MIP)}$. It will always be the case that $\text{lowbound (MIP)} \geq \text{lowbound (SIP)}$. This follows from the fact that the MIP imbedded problem is the same as the SIP imbedded problem except that it incorporates more constraints. Nothing can be said, in general, about the relation between the initial lower bound of the LIP scheme and the other schemes.

The synthesis of the LIP and SIP schemes was suggested since, for both of these schemes, there is a one-to-one correspondence between the variable X_{ij} and the flow in a particular arc of the network representation. This correspondence does not hold in the MIP scheme since any node representing a course has one arc going to it with a flow of one, but several arcs leaving it going to nodes representing the same time, each with a maximum capacity of one and a minimum capacity of zero. These facts can be observed in Figure 9 of Chapter III.

The general method of synthesis is exhibited in Figure 18.

Even though initially the general synthesis procedure may seem to be inefficient in the sense that two sets of lower bounding problems must be solved at each level of computation, the advantages that are associated with this approach are twofold. First, it is possible to obtain a feasible solution each time a lower bounding problem is

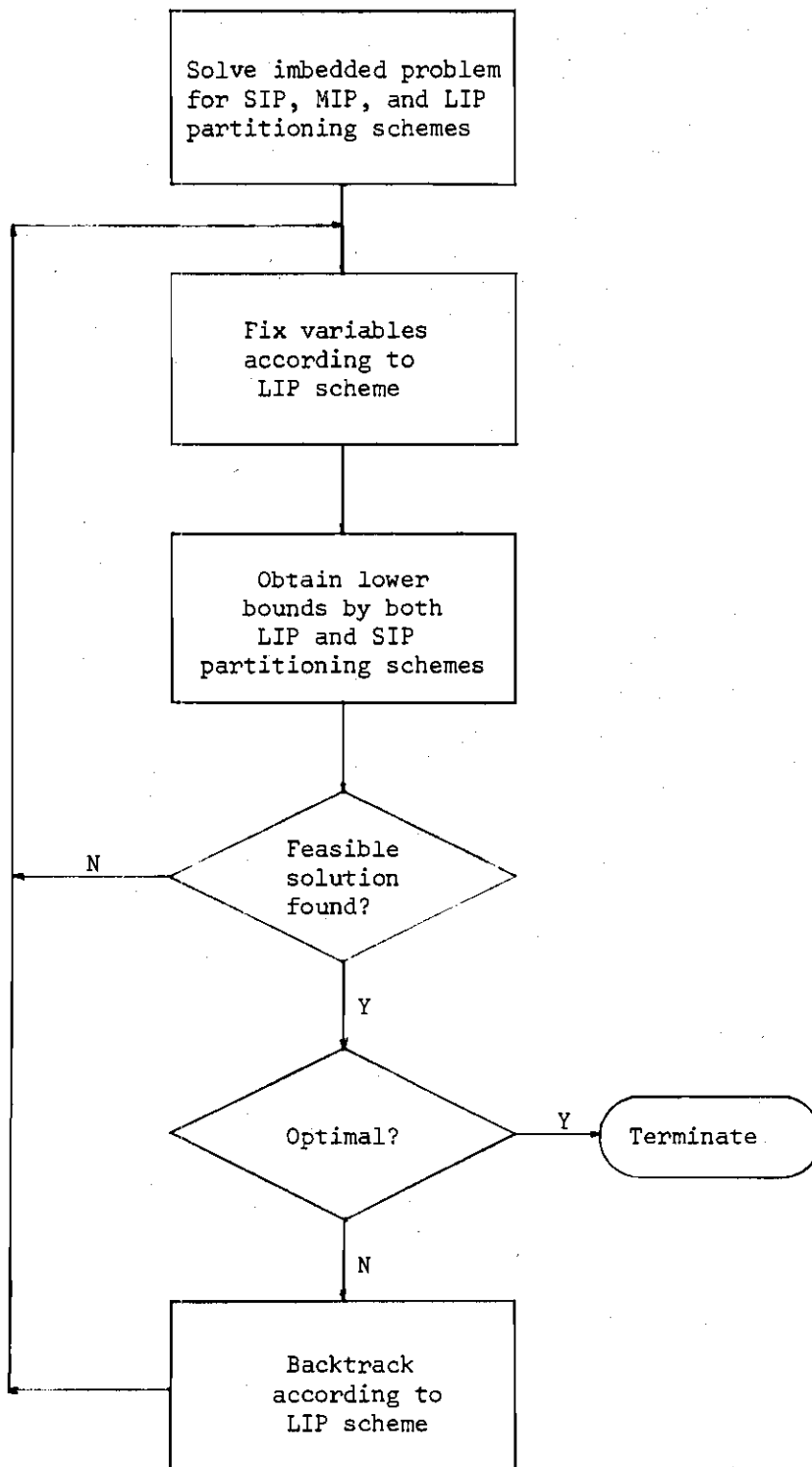


Figure 18. General Synthesis Scheme

solved. Since the lower bounding problems are different in structure and since the variables are fixed according to the LIP scheme, the solutions obtained by solving the LIP lower bounding procedure are feasible to a large set of additional constraints of the SIP scheme, and vice versa. Second, the large amount of computational effort associated with backtracking in the LIP scheme may be eliminated due to the tighter lower bounds obtained with the SIP scheme.

CHAPTER V

CONCLUSIONS AND RECOMMENDATIONS

Although the algorithm presented in Chapter II was developed to solve general problems which have imbedded in their constraint set a subset of constraints whose coefficient matrix possesses the unimodular property, the application discussed in Chapters III and IV was very specific in nature, namely, the course scheduling problem. In the development of the algorithm it was specified that the original integer program be converted to a zero-one integer programming problem. The main purpose for this transformation was the simplification it allowed in partitioning the solution space of the imbedded problem. Since it is the solution space of the imbedded problem that is partitioned, it suffices to transform the imbedded problem to a zero-one integer programming problem, and when feasibility of the additional constraint set is to be determined, the zero-one variables may be transformed back to the original variables.

For situations in which the imbedded problem is of the form:

$$\text{minimize} \quad \sum_{i=1}^m \sum_{j=1}^n C_{ij} X_{ij}$$

subject to:

$$\sum_{i=1}^m X_{ij} = A_j \quad \forall i$$

$$\sum_{j=1}^n X_{ij} \leq B_i \quad \forall j$$

$$x_{ij} \geq 0 \quad \forall i, j$$

which is a transportation problem, or for more general imbedded problems which have an equivalent network formulation, the imbedded problem may be readily transformed to a zero-one integer programming problem by the following procedure:

define:

$$Y_{ij} = 0 \text{ or } 1$$

such that:

$$\sum_{i=1}^m Y_{ij} = 1 \quad \forall j$$

and let:

$$X_{ij} = \sum_{K=\alpha_j}^{\beta_j} Y_{iK} \quad i=1, m$$

where:

$$\alpha_j = \sum_{L=0}^{j-1} A_L + 1 \quad \text{with} \quad A_0 = 0 \quad \forall j$$

$$\beta_j = \sum_{L=0}^j A_L \quad \forall j$$

Substituting for x_{ij} in the original problem yields:

$$\text{minimize} \quad \sum_{i=1}^m \sum_{j=1}^n C_{ij} \sum_{K=\alpha_j}^{\beta_j} Y_{iK}$$

subject to:

$$(1) \quad \sum_{i=1} \sum_{K=\alpha_j}^{\beta_j} Y_{iK} = A_j \quad j=1,n$$

$$(2) \quad \sum_{j=1}^n \sum_{K=\alpha_j}^{\beta_j} Y_{iK} \leq B_i \quad i=1,m$$

$$Y_{iK} \geq 0 \quad \forall i,K$$

Expanding the above yields:

$$\text{minimize} \quad \sum_{i=1}^m \sum_{j=1}^n C_{ij} \sum_{K=\alpha_j}^{\beta_i} Y_{iK}$$

subject to:

$$(1) \quad \sum_{i=1}^n Y_{iK} = 1 \quad K = 1,N = \sum_{j=1}^n A_j$$

$$(2) \quad \sum_{K=1}^N Y_{iK} \leq B_i \quad i=1,m$$

$$(3) \quad Y_{iK} \geq 0 \quad i=1,m; K=1,N$$

The resulting problem in the Y_{ij} 's still possesses a constraint coefficient matrix which is unimodular. By constraint set (1) the zero-one

restriction on the Y_{ij} 's will automatically be satisfied.

In terms of the network equivalent to the above problem, the only difference is that the number of nodes and arcs have been increased over the original network representation. Since solution methods for network problems are extremely efficient, the suggested transformation can be made providing that the number of arcs and nodes that must be adjoined to the problem is not excessive.

It is recommended that further computational experience should be obtained with the algorithm presented in this thesis for different types of problems which possess imbedded unimodular constraints. Furthermore, for other types of applications, attention should be paid to possible additional advantages that may be derived from the structure of the problem.

As an extension to the application area presented in this thesis, it is recommended that the concepts developed be implemented for solution to the university-wide course scheduling problem. In Chapter III it was made clear that the particular course scheduling problem formulated represented a problem faced by the School of Industrial and Systems Engineering in scheduling its courses. This subsystem of ISEMIS will not create conflicts for students providing that they are only required to take the courses offered by that school. If, however, a student is taking courses from several schools within the university simultaneously, as is normally the case, then it is quite likely that conflicts will arise because courses offered by different schools of the university which he must take simultaneously could be offered at

the same time. Consequently, the problem of scheduling courses should ideally consider the interdependencies between the various schools.

A formulation of the university-wide course scheduling problem can be developed along similar lines as the course scheduling problem for a single school, namely, define the following:

$$X_{ijP} = \begin{cases} 1 & \text{if course } j \text{ of school } P \text{ is offered at time } i \\ 0 & \text{otherwise} \end{cases}$$

n_P = Number of courses to be scheduled for school P.

D_{KP} = Set of courses with forecasted enrollment $\leq U_P^K$ in school P.

U_P^K = Maximum capacity of classrooms in group K of school P.

M_{iP}^K = Number of classrooms available in group K at time i for school P.

W_{KP} = Set of courses taught by faculty member K of school P.

L_{KP} = Set of courses which conflict within school P.

M_{1P} = Number of classroom groupings in school P.

M_{2P} = Number of faculty in school P.

M_{3P} = Number of course conflict groups in school P.

θ_K = Kth set of inter-school courses which, if taught simultaneously, would create conflicts for students.

With the above definitions, the university-wide scheduling problem may be stated as:

$$\text{minimize} \quad \sum_{P=1}^{\tau} \sum_{J=1}^{n_P} \sum_{i=1}^m C_{ijP} X_{ijP}$$

subject to:

$$\begin{array}{l}
 (1) \quad \sum_{i=1}^m X_{ijP} = 1 \quad j=1,n \\
 (2) \quad \sum_{j \in D_{KP}} X_{ijP} \leq M_{iP}^K \quad i=1,m; K=1,m_{1P} \\
 (3) \quad \sum_{j \in W_{KP}} X_{ijP} \leq 1 \quad i=1,m; K=1,m_{2P} \\
 (4) \quad \sum_{j \in L_{KP}} X_{ijP} \leq 1 \quad i=1,m; K=1,m_{3P} \\
 (5) \quad \sum_{(J,P) \in \theta_K} X_{ijP} \leq 1 \quad i=1,m; K=1,m_4 \\
 X_{ijP} = \text{binary} \quad \forall i,j,P
 \end{array}
 \left. \vphantom{\begin{array}{l} (1) \\ (2) \\ (3) \\ (4) \\ (5) \end{array}} \right\} P = 1,r$$

As can be noted, constraint sets (1) through (4) in the above formulation are identical to the constraint set of the Coursched subsystem, and that there are as many independent constraint sets of that form as there are schools within the university. The constraint set represented by (5) are those which indicate the interdependency between the courses offered in different schools in the sense that if they are taught at the same time, conflicts would be created for various segments of the student population.

Figure 19 represents, in a conceptual manner, the structure of the constraint coefficient matrix of the constraints defined on the

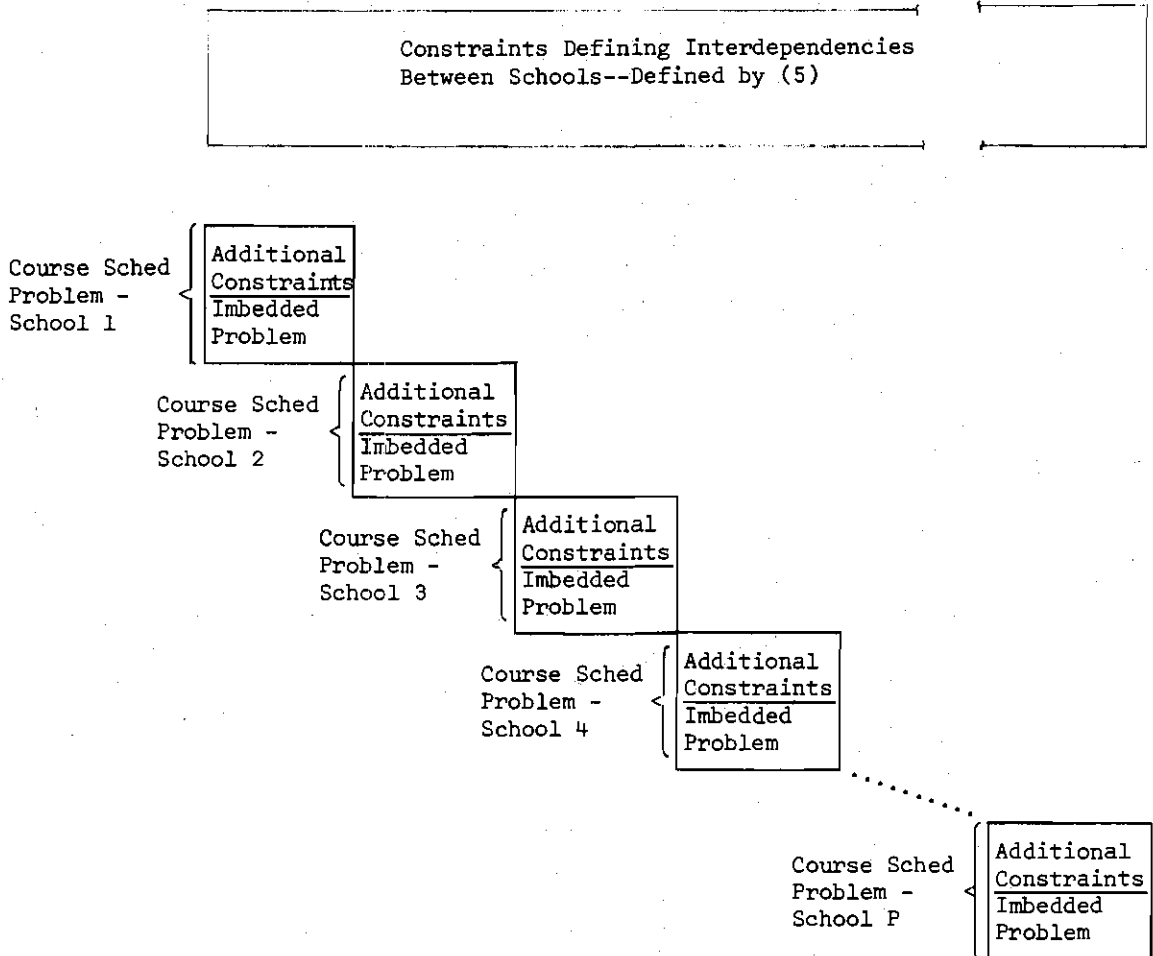


Figure 19. Conceptual Structure of University-Wide Course Scheduling Problem

previous pages. As can be noted, it is a block diagonal type structure, the individual blocks of constraints being linked together by the constraints defined by (5). One immediate solution procedure which presents itself would consist of adjoining the additional constraints for each school with the constraints defining interdependence between the schools. The resulting problem would then be of the type applicable directly to the solution algorithm presented in this thesis. This can be noted by observing that, in Figure 19, the remaining constraints of the block diagonal elements would correspond to the individual imbedded problems, and consequently the constraint coefficient matrix containing the remains of each block of constraints would possess the unimodular property.

Another approach to the solution of the university-wide course scheduling model which is different in concept to the previous can be considered. For ease of exposition, consider a conceptual matrix representation of the structure defined in Figure 19, namely:

$$\text{minimize} \quad C_1 X_1 + C_2 X_2 + C_3 X_3 + \cdots + C_P X_P$$

subject to:

$$A_1 X_1 + A_2 X_2 + A_3 X_3 + \dots + A_P X_P \leq b_0$$

$$\begin{bmatrix} B_1 \\ U_1 \end{bmatrix} X_1 \leq b_1$$

$$\begin{bmatrix} B_2 \\ U_2 \end{bmatrix} X_2 \leq b_2$$

$$\begin{bmatrix} B_3 \\ U_3 \end{bmatrix} X_3 \leq b_3$$

$$\begin{bmatrix} B_P \\ U_P \end{bmatrix} X_P \leq b_P$$

X_i = Vectors with binary components $i=1,P$

where:

X_i - Vector of decision variables for school i .

C_i - Row vector of cost coefficients for school i .

B_i - Coefficient matrix of additional constraints associated with the course scheduling problem formulation for school i .

U_i - Coefficient matrix of imbedded constraints associated with the course scheduling problem formulation for school i .

A_i - Portion of overall constraints coefficient matrix corresponding to the decision vector X_i .

b_0 - Right-hand side vector of overall constraints.

b_i - Right-hand side vector associated with course scheduling problem for school i .

With the above representation of the university-wide course scheduling

problem, a solution procedure combining some of the ideas of stage-wise decision making and implicit enumeration can be devised. Consider a P stage optimization problem, in which the first stage consists of the problem:

$$\text{minimize } C_1 X_1$$

subject to:

$$A_1 X_1 \leq b_0$$

$$\begin{bmatrix} B_1 \\ U_1 \end{bmatrix} X_1 \leq b_1$$

X_1 = Vector with binary components.

The above is the course scheduling problem for school one with an enlarged set of additional constraints. The methodology necessary to solve it has been developed in this thesis. One modification necessary for the stage-wise solution of the university-wide problem is that in the construction of the BHT, lowbound0 and lowbound1 should be modified by the addition of a constant, this constant is calculated by solving the problem:

$$\text{minimize} \quad C_2 X_2 + C_3 X_3 + \dots + C_P X_P$$

subject to:

$$\begin{aligned} U_2 X_2 &= b_2^u \\ U_3 X_3 &= b_2^u \\ &\dots \\ b_P X_P &= b_P^u \end{aligned}$$

X_i = Vector with binary components

where:

b_i^u - Right-hand side vector associated with the imbedded problem for school i .

The constant referred to earlier is the objective function value obtained from solving the above problem. This modification is necessary in order to account for the remaining stages of computation after the problem associated with the present stage has been solved.

Denoting the optimal solution vector of stage i by X_i^* , the general problem associated with the q th stage is:

$$\text{minimize} \quad C_q X_q$$

subject to:

$$A_q X_q \leq b_o - \sum_{i=1}^{q-1} A_i X_i^*$$

$$\begin{bmatrix} B \\ q \\ U \\ q \end{bmatrix} X_q \leq b_q$$

X_q = Vector with binary components.

The constant that must be added to lowbound0 and lowbound1 during the computations of this stage is:

$$\sum_{i=1}^{q-1} C_i X_i^* + K_q$$

where K_q is the objective function value obtained by solving the problem:

$$\text{minimize} \quad \sum_{i=q+1}^P C_i X_i$$

subject to:

$$U_i X_i = b_i^u \quad i=q+1, P$$

X_i = Vector with binary components $i=q+1, P$

The BHT itself will be constructed in a continuous sequential manner as the various stages are considered during the computation. If backtracking is necessary at some level of computation, care must be exercised in defining the constant terms that are added to lowbound0 and lowbound1

once another branch has been found to explore further. For example, if it is necessary to backtrack from a level of computation encountered in the n th stage of computation, and if the backtracking terminates in the $n-1$ th stage, the constants added to `lowbound0` and `lowbound1` must be changed from those being utilized during the n th stage to the appropriate constants associated with the $n-1$ st stage.

It is recommended that the `Coursched` subsystem of ISEMIS be generalized to a university-wide model along the general concepts presented in the previous sections. It is also recommended that the computational aspects of the stage-wise solution suggested be explored with respect to the organization of the computations and the efficiencies obtainable.

BIBLIOGRAPHY

1. Bakes, M. D., "Solution of Special Linear Programming Problems with Additional Constraints," *Operational Research Quarterly*, Vol. 17, No. 4.
2. Balas, E., "A Note on the Branch and Bound Principle," *Operations Research*, Vol. 16, No. 2.
3. Balas, E., "An Additive Algorithm for Solving Linear Programs with Zero-One Variables," *Operations Research*, Vol. 13, No. 4.
4. Balas, E., *Duality in Discrete Programming: IV. Applications*, Management Sciences Research Report No. 145, Carnegie Mellon University, Pittsburgh, Pennsylvania.
5. Balas, E., *Duality in Discrete Programming*, Technical Report No. 67-5, Department of Operations Research, Stanford University, Stanford, California.
6. Balas, E., *Minimax and Duality for Linear and Nonlinear Mixed-Integer Programming*, Working Paper 9-69-7, Carnegie Mellon University, Pittsburgh, Pennsylvania.
7. Benders, J. F., "Partitioning Procedures for Solving Mixed-Variables Programming Problems," *Numerische Mathematik*, Vol. 4, pp. 253-261.
8. Bozoki, George, *Minimum Cost Multicommodity Network Flows*, Ph.D. Dissertation, Purdue University, LaFayette, Indiana, 1969.
9. Bray, T. A., and Witzgall, T., "Algorithm 336-Netflow," *Communication of the ACM*, Vol. 11, No. 9.
10. Charnes, A., and Cooper, W. W., *Management Models and Industrial Applications of Linear Programming - Vol. II*, John Wiley & Sons, Inc., New York, New York, 1961.
11. Charnes, A., and Lemke, C. E., *Multi-Copy Generalized Networks and Multipage Programs*, Math. Rep. No. 41, Rensselaer Polytechnic Institute, Troy, New York.
12. Dantzig, G. B., *Linear Programming and Extensions*, Princeton University Press, Princeton, New Jersey, 1963.

13. Dantzig, G. B., and Wolfe, P., "The Decomposition Algorithm for Linear Programs," *Econometrica*, Vol. 29, No. 4.
14. Ford, R. L., and Fulkerson, D. R., *Flows in Networks*, Princeton University Press, Princeton, New Jersey, 1962.
15. Geoffrion, A. M., "Integer Programming by Implicit Enumeration and Balas' Method," *SIAM Review*, Vol. 9, No. 2.
16. Gomory, R. E., "Outline of an Algorithm for Integer Solutions to Linear Programs," *Bulletin of the American Mathematical Society*, Vol. 64, 1958.
17. Gomory, R. E., *All-Integer Integer Programming Algorithm*, Research Report RC-189, IBM Research Center, Yorktown Heights, New York.
18. Hoffman, A. J., and Kruskal, J. B., "Integral Boundary Points of Convex Polyhedra," in: *Linear Inequalities and Related Systems*, Annals of Mathematics Study No. 38, Princeton University Press, Princeton, New Jersey, 1956.
19. Hu, T. C., *Integer Programming and Network Flows*, Addison-Wesley Publishing Co., Reading, Massachusetts, 1969.
20. Young, R. D., "A Simplified Primal Call-Integer Programming Algorithm," *Operations Research*, Vol. 16, No. 4.

VITA

The author was born in Haaren, Holland, on July 21, 1944. He received his primary education in Holland, Argentina, and Venezuela.

In 1958 the author came to the United States to further his education. He received his high school education at Camden Military Academy, Camden, South Carolina, from which he graduated in June, 1961.

In June, 1965, the author received a B.S. degree in Industrial Engineering, with honors, from Clemson University. He then attended the Georgia Institute of Technology from which he received the M.S.I.E. degree in June, 1968, and the Ph.D. degree in September, 1970.

While attending the Georgia Institute of Technology, the author joined the faculty, holding the rank of Instructor and, later, Assistant Professor in the School of Industrial and Systems Engineering.

The author's industrial experience includes part-time work with the computer application group of the International Paper Company, and with the Corporate Operations Research Group of E. I. DuPont de Nemours & Co., Inc. He will serve as a consultant to the Organization for Economic Co-operation and Development (OECD) for one year starting in September, 1970. This activity will place him as a Visiting Assistant Professor at the Middle East Technical University in Ankara, Turkey.

The author is a member of the Operations Research Society of America, the Institute of Management Sciences, the Association for

Computing Machinery, the Society of Sigma Xi, and Alpha Pi Mu. He is married and has no children.