INTEGER PROGRAMMING BY GROUP THEORY: SOME COMPUTATIONAL RESULTS

Harvey M. Salkin, et al

Case Western Reserve University

Prepared for:

Office of Naval Research

January 1975

INTEGER PROGRAMMING BY GROUP THEORY: SOME COMPUTATIONAL RESULTS*

Harvey M. Salkin
Susumu Morito

Technical Memorandum #341

January 1975

* Part of this research was supported by the Office of Naval Research contract no. N00014-67-A-0404-0010.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: Technical Memorandum #341	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): "Integer Programming by Group Theory: Some Computational Results"	5. TYPE OF REPORT & PERIOD COVERED: 1974
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): Harvey M. Salkin Susumu Morito	8. CONTRACT OR GRANT NUMBER(s): N00014-67-A-0404-0010
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Case Western Reserve University Cleveland, Ohio 44106	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Office of Naval Research Arlington, Virginia 22217	12. REPORT DATE: January, 1975
	13. NUMBER OF PAGES: 37
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report): unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

unlimited

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

18. SUPPLEMENTARY NOTES

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

Integer Programming	Branch and Bound
Group Theory	Computational Results
Dynamic Programming	Mathematical Programming

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

A group theoretic algorithm for the integer program has been computer programmed and tested. It basically consists of a linear programming algorithm, a routine which converts the (relaxed) integer program to a group minimization problem (over the fractional column group or the isomorphic factor group attained via Smith's Normal Form), solving the group problem by dynamic programming or by a shortest path algorithm, and when necessary, uses a branch and bound procedure. Details and computational results are given. Future work regarding other computational strategies available to group theoretic algorithms is also included.

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

PRICES SUBJECT TO CHANGE

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

CONTENTS

ABSTRACT

A group theoretic algorithm for the integer program has been computer programmed and tested. It basically consists of a linear programming algorithm, a routine which converts the (relaxed) integer program to a group minimization problem (over the fractional column group or the isomorphic factor group attained via Smith's Normal Form), solving the group problem by dynamic programming or by a shortest path algorithm, and when necessary, uses a branch and bound procedure. Details and computational results are given. Future work regarding other computational strategies available to group theoretic algorithms is also included.

1. INTRODUCTION

In 1960 Ralph Gomory [G1] indicated that the coefficient vectors of the inequalities derived in his dual fractional algorithm form an abelian group, which can have at most D elements, where D is the absolute value of the determinant of the current linear programming basis. Based on these results Gomory [G2] in 1965 showed that by relaxing nonnegativity, but not integrality constraints on certain variables, an integer program may be transformed to one whose columns of constraint coefficients, and the right hand side are elements of an abelian group. If this group problem is solved and its solution yields nonnegative values for the variables of the original problem, then the integer program has been solved. The group problem can be treated as an integer program with one constraint (i.e., a knapsack problem) or as a network problem, where a shortest route is desired. In this paper, we will discuss solution strategies for the group minimization problem . Some computational experience will be presented. Future work, now being developed, will also be mentioned.

Consider the integer program

P1 maximize cx

subject to $Ax = b$

$x \geq 0$ and integer,

where $A = (A;I)$ (i.e., original problem is of the form $A'x' \leq b$) is an m by $m + n$ matrix whose rank is m, I is an m identity matrix, $c = (c', 0)$, and $x = (x', s')$, where s are slack variables. Now suppose B is a basis whose columns are from A, and that we rearrange the terms in the previous problem so that it is the same as

P2 maximize $c_B x_B + c_N x_N$

subject to $B\,x_B + N\,x_N = b$

$x_B \geq 0$, $x_N \geq 0$ and integer,

where x_B (x_N) are the basic (nonbasic) variables associated with B (N), and the costs corresponding to the basic (nonbasic) variables are c_B (C_N). Using the integrality of x_B we get the equivalent problem P3.

P3 maximize $c_B B^{-1} b - (c_B B^{-1} N - c_N)\, x_N$

subject to $B^{-1} N\, x_N \equiv B^{-1} b \pmod 1$

$x_N \geq 0$, integer

and $x_B = B^{-1} b - B^{-1} N\, x_N \geq 0.$

Suppose that we choose a basis B which yields a dual feasible solution, that is $\bar{c} = c_B B^{-1} N - c_N \geq 0$; for example, the optimal linear programming basis may be selected. Dropping the constant term $c_B B^{-1} b$, converting the problem into minimization format, and omitting the nonnegativity requirements x_B, yields

P4 minimize $\bar{c}\, x_N$

subject to $B^{-1} N\, x_N \equiv B^{-1} b \pmod 1$

and $x_N \geq 0$, integer.

This problem is equivalent to

P5(GMP) minimize $\sum_{j=1}^{n} \bar{c}_j\, x_{J(j)}$

subject to $\sum_{j=1}^{n} \bar{\alpha}_j\, x_{J(j)} \equiv \bar{\alpha}_0 \pmod 1$ (1)

$x_{J(j)} \geq 0$ and integer $(j = 1, \ldots, n)$,

where $\bar{c} = (\bar{c}_j) \geq 0$, column $\bar{\alpha}_j$ $(j = 0, \ldots, n)$ is the fractional parts of the

j^{th} column of $B^{-1}N$ and satisfies $0 \leq \bar{\alpha}_j < e$ (a column of ones), and $x_N = (x_{J(j)})$ is the j^{th} $(1 \leq j \leq n)$ nonbasic variable. Problem P5 is referred to as the group minimization problem (GMP) over the fractional column group.

Denote the set of vectors generated by repeated additions (modulo 1) of the n $\bar{\alpha}_j$'s in P5 as $G(\bar{\alpha})$. Then it can be shown [S1], either $G(\bar{\alpha})$ is a cyclic group or it can be expressed as a direct sum of cyclic subgroups. Moreover, the order of $G(\bar{\alpha})$ equals D or it divides D. Also, it equals D whenever A contains an identity matrix.

To solve GMP, we can think of the equalities (1) as one constraint and apply a dynamic programming procedure. In particular, for k = 1, ..., n let

$$f(k,g) \triangleq \text{minimize} \quad \sum_{j=1}^{k} \bar{c}_j \, x_{J(j)}$$

$$\text{subject to} \quad \sum_{j=1}^{k} \bar{\alpha}_j \, x_{J(j)} \equiv g \pmod 1$$

$$x_{J(j)} \geq 0 \text{ and integer } (j=1, \ldots, k).$$

That is, f(k,g) is the minimal value of the objective function to the GMP using the first k variables and with right hand side $\bar{\alpha}_o$ replaced by g, and $f(n,\bar{\alpha}_o)$ is the value of the minimal solution to the GMP. The recursive relationship (2) allows us to compute f(k,g) for all k = 1, ..., n and g in $G(\bar{\alpha})$ provided that every $\bar{\alpha}_k$ generates the group $G(\bar{\alpha})$.

$$f(k,g) = \text{minimum } \{f(k-1,g), \; \bar{c}_k + f(k,g-\bar{\alpha}_k)\} \tag{2}$$

To retrieve the solution, we define j(k,g) to be the index of the last variable used in making up f(k,g). Thus, j(1,g) = 1 for all g and for $k \geq 2$ define

$$j(k,g) = j(k-1,g) \quad \text{if } x_{J(k)} = 0 \text{ or } f(k,g) = f(k-1,g)$$

and $$j(k,g) = k \quad \text{if } x_{J(k)} \geq 1 \text{ or } f(k,g) = \bar{c}_k + f(k,g-\bar{\alpha}_k).$$

If each $\bar{\alpha}_j$ does not generate the group $G(\bar{\alpha})$, we may use a modified procedure suggested by T.C. Hu [H1]. In particular, suppose $\bar{\alpha}_k$ $(k \geq 2)$ generates a

subgroup $G(\bar{\alpha}_k)$ of order $d < D$. Expression (2) will give $f(k, \beta\bar{\alpha}_k)$ for $\beta = 1,\ldots,d$, but suppose we need $f(k,g)$ for an element g which is not in $G(\bar{\alpha}_k)$. We have the value of $f(k-1,g)$, and need the value of $f(k,g-\bar{\alpha}_k)$ to calculate $f(k,g)$. So, we temporarily assume that $f(k,g-\bar{\alpha}_k) = f(k-1,g-\bar{\alpha}_k)$, which is, at worst, an overestimate. Note that $f(k-1,g-\bar{\alpha}_k)$ is known. We use the following modified recursive relationship.

$$f(k,g + \beta\bar{\alpha}_k) = \text{minimum } \{\underbrace{f(k-1,g + \beta\bar{\alpha}_k)}_{\text{known}}, \underbrace{\bar{c}_k + f(k,g + \beta\bar{\alpha}_k-\bar{\alpha}_k)}_{\text{tentatively estimated}}\} \quad (3)$$

Starting with $\beta = 1$, we obtain a new estimate for $f(k,g)$ when $\beta = d$, and if this value agrees with the original one it can be shown to be the correct value ([S1]). If it differs, the process continues. This process converges to the correct value sometime between step d and step 2d inclusively.

If the optimal solution, x_N, for the group minimization problem yields nonnegative basic variables, x_B, then (x_B,x_N) is an optimal solution to the original IP problem. Specifically, given the solution for GMP, x_N, if

$$x_B = B^{-1}b - B^{-1}N\, x_N \geq 0$$

then we have solved the IP problem. If, on the other hand, one or more of the basic variables turns out to be negative, we have to seek the smallest value of $\bar{c}x_N$ for which $x_B \geq 0$ and x_N satisfies the constraints of the GMP.

White [W1] developed an algorithm to find the k^{th} best solution to the group problem using a set of recursive relationships. This is rather complicated and difficult to implement. Instead of White's approach, we use a branch and bound enumeration which is similar to the Dakin [D1] variation. The scheme presented here inspects the integer solutions to the group problem by successively adding constraints of the form $x_{J(j)} \geq K$ (j=1, ..., n), where K starts at 0 and is increased by 1. In terms of tree, a node corresponds to a vector x_N with a greater than or equal to inequality acting on each variable, a branch to introducing the inequality in such a way that two nodes are joined

by a branch whenever the inequality acting on one particular variable has been incremented by 1 and the others are the same. A detailed explanation of the procedure is in [S1].

Using Smith's Normal Form (SNF) [H1], [M1] we can construct a different group minimization problem. In particular, SNF diagonalizes a given m by m nonsingular integral (basis) matrix B, such that $\hat{B} = RBC$, where $\hat{B}$ is a diagonal matrix of the form

$$\hat{B} = \begin{pmatrix} \varepsilon_1 & & \bigcirc \\ & \varepsilon_2 & \\ \bigcirc & & \ddots \\ & & & \varepsilon_m \end{pmatrix},$$

the ε_i are positive integers (i=1,...,m), ε_i divides ε_{i-1} (i=1,...,m-1), and $|\det. R| = |\det. C| = 1$. This can be used to construct the group minimization problem (see[S1])

P6 (FGMP) minimize $\sum_{j=1}^{n} \bar{c}_j x_{J(j)}$

subject to $\sum_{j=1}^{n} \lambda_j x_{J(j)} \equiv \lambda_o \mod \begin{pmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_m \end{pmatrix}$

$x_{J(j)} \geq 0$ and integer (j=1, ..., n),

where λ_j is the j^{th} column of RN with each component i reduced modulo ε_i, λ_o is the vector Rb with each element reduced modulo ε_i, and $\hat{B} = RBC$. The set $G(\lambda)$ generated by the λ_j's (j=1, ..., n) is an abelian group which can be shown to be isomorphic to $G(\bar{\alpha})$. The details are in [S1].

Note that if $\varepsilon_i = 1$, $\sum_{j=1}^{n} \lambda_{ij} x_{J(j)} \equiv \lambda_{io} \pmod{\varepsilon_i}$ is satisfied for any integer solution. Here λ_{ij} is the i^{th} element of the j^{th} column of RN. This is because all elements of the constraint matrix are integer, as well as the right

hand side. In this case, the i^{th} congruence equation contains only zeros and thus can be omitted. This means that all rows which have $\varepsilon_i=1$ can be omitted from the constraints. We will denote the number of effective rows in FGMP as m_F, which is m minus the number of rows with $\varepsilon_i=1$. Gorry, Northrup, and Shapiro [G3] claim that m_F is usually 1 to 5 regardless of the value of m.

2. THE GROUP MINIMIZATION ALGORITHM

The intension of this paper is to expose the efficacy of various existing options. Specifically, we have the three major options in this algorithm listed in Table 1. Moreover, there are many strategies for each option. The natural

A. Form of the group minimization problem	(1) Regular group minimization problem (using fractional columns		(2) Factor group minimization problem (using SNF)
B. How to solve the group minimization problem	(1) Dynamic programming algorithm (DP)		(2) Shortest path algorithm (SP)
C. How to solve the integer program if the group problem does not solve it.	(1) Branch and bound	(2) Dynamic programming	(3) k^{th} shortest path algorithm

TABLE 1

question is which of the various possibilities appears to be most efficient. Some computational work in this direction has already appeared in the literature ([G1, S3, W1, G3, G4, G5, H4]). An exposition of that work can be found in [S1]. Table 2 summarizes the strategies used and numerical results. Even though some studies attempt to compare possible strategies they are, at best, quite incomplete. The work by Gorry, Northrup and Shapiro [G3] is somewhat more extensive, but the comparisons of various strategies is not included.

Paper	Algorithm				Computational Experience					
	Form of the Group Problem	Solving Algorithm	When $x_B \gtrless 0$	Remarks	m x n	D	r*	Computation time (Sec.)	Language & machine	Remarks
W. W. White (1966) [W1]	G M P	DP Procedure for finding the k-th best solution. It is shown that if an optimal integer solution exists, it can be found by finding the k-th best solution.		White's algorithm is equivalent to finding k-ch shortest path through a specially constructed network.	2x4 ⋮ 21x56	32 ⋮ 2856	1 ⋮	6.3 ⋮ 273.1	FORTRAN IV IBM 360/65	The efficiency of this algorithm depends crucially on D. Rescaling of rows and columns suggested in the paper.
J.F. Shapiro (1968) [S3]	F G M P via S N F	Specially structured shortest path algorithm.	Branch and bound enumeration.	The algorithmic extension of Gomory's DP method and inclusion of implicit enumeration scheme when $x_B \gtrless 0$.	3x7 ⋮ 50x65	3 ⋮ 2856	1 ⋮	0.81 ⋮ 20.20	FORTRAN IBM 360/65	Computation times increase almost linearly as D increases. A zero-one shortest path algorithm is used for zero-one problems.
F. Glover (1969) [G4]	Not treated. (But assume FGMP)	Dual-like enumeration procedure (not DP type).	Not treated.	A dual method in that optimal solutions are generated for a sequence of right hand sides, until a feasible solution is found.	1x50 ⋮ 1x1500	100 ⋮ 4500	1	0.81 ⋮ 12.11	FORTRAN IV CDC 6400	Accelerated version of Glover's algorithm works good and computation times increase at a considerably more favorable rate than nD.
D.S. Chen & S. Zionts (1972) [C1]	Not treated. (But assume FGMP)	Modified shortest path algorithms, i.e., 1) Floyd 2) Two-way 3) Modified Dantzig	Not treated.	A scheme for modifying and simplifying existing shortest path algorithms by exploiting the special features of the network.	1x2 ⋮ 1x99	20 ⋮ 60 ⋮ 100	1	<1	FORTRAN IV CDC 6-00	The two-way method dominates when ratio of n to D is moderate and large, while Shapiro's algorithm is most efficient for the small ratio.
G.L. Hefley & M.E. Thomas (1972) [H4]	Not explicitly mentioned (But assume FGMP)	Decomposition algorithm 1) Subproblems solved by Gomory DP algorithm 2) Linking algorithm is enumeration type.	Not treated.	This algorithm decomposes the group problem into subproblems defined on subgroups of the original problem.	4x14 ⋮ 14x68	32 ⋮ 2000	2 ⋮ 6	0.11 ⋮ 31.65	PL-1 IBM 360/65	Savings are substantial when r is large, say r = 5 or 6. For small r this is not the case.
G.A. Gorry, J.F. Shapiro & L.A.Wolsey (1973) [G5]	F G M P via S N F	Specially structured shortest path algorithm	Branch and bound enumeration.	Tries to establish that group theoretic IP methods are easy to construct and effective to use.	14x32 ⋮ 313x482 ⋮ 176x2385	10 ⋮ 81600	1 ⋮ 6	0.42 ⋮ 191.3	FORTRAN UNIVAC 1108 IBM 360 /65,85	Computational experience with this code appears excellent even for large-scale (in terms of D and m x n) real life problems.

*r is the number of cyclic subgroups treated.

TABLE 2. GROUP MINIMIZATION ALGORITHMS: COMPUTATIONAL EXPERIENCE

A computer system with several subroutine options, for solving an integer program using group minimization techniques has been developed. The general procedures of our code appears in Figure 1 and the detailed flow charts of major subprograms are in later pages. We now discuss the composition of the computer system. The paranthesized letter adjacent to each heading agrees with Figure 1.

(a) The LP Code

The linear programming algorithm used in this code is a variation of the revised simplex method applied to the dual. The basis inverse is kept in product form and a basis reinversion procedure is used to reduce roundoff errors. The original program was written by Salkin and Spielberg [S2] and was later modified for and incorporated in the set covering code SCA 1 [S2].

Because it uses the dual method, the algorithm requires dual feasibility. If the primal is feasible, the solution is optimal. It turns out that a basic dual feasible solution can always be obtained by using the complementing variables $\bar{x}_j = u_j - x_j$, for those variables with negative costs. Here u_j is an integral upper bound for x_j. Using this technique we can avoid a phase I procedure.

(b) Construction of the Group Problem

This part of the program converts the relaxed integer program to a group minimization problem. The resulting problem may be either the one over the fractional column group (GMP)(b1), or the one over the isomorphic factor group (FGMP)(b3) attained via Smith's normal form (SNF)(b2). More specifically, given the LP optimal solution GMP converts problem P1 into P5, whereas FGMP converts P1 into P6 with the diagonalized LP optimal basis matrix produced by SNF.

Clearly it is more time consuming to diagonalize the basis matrix, but the number of rows of FGMP usually is much less than that of GMP. This allows a simpler bookkeeping scheme to keep track of the problem and less computer storage requirements. A trade-off exists between SNF conversion time and the

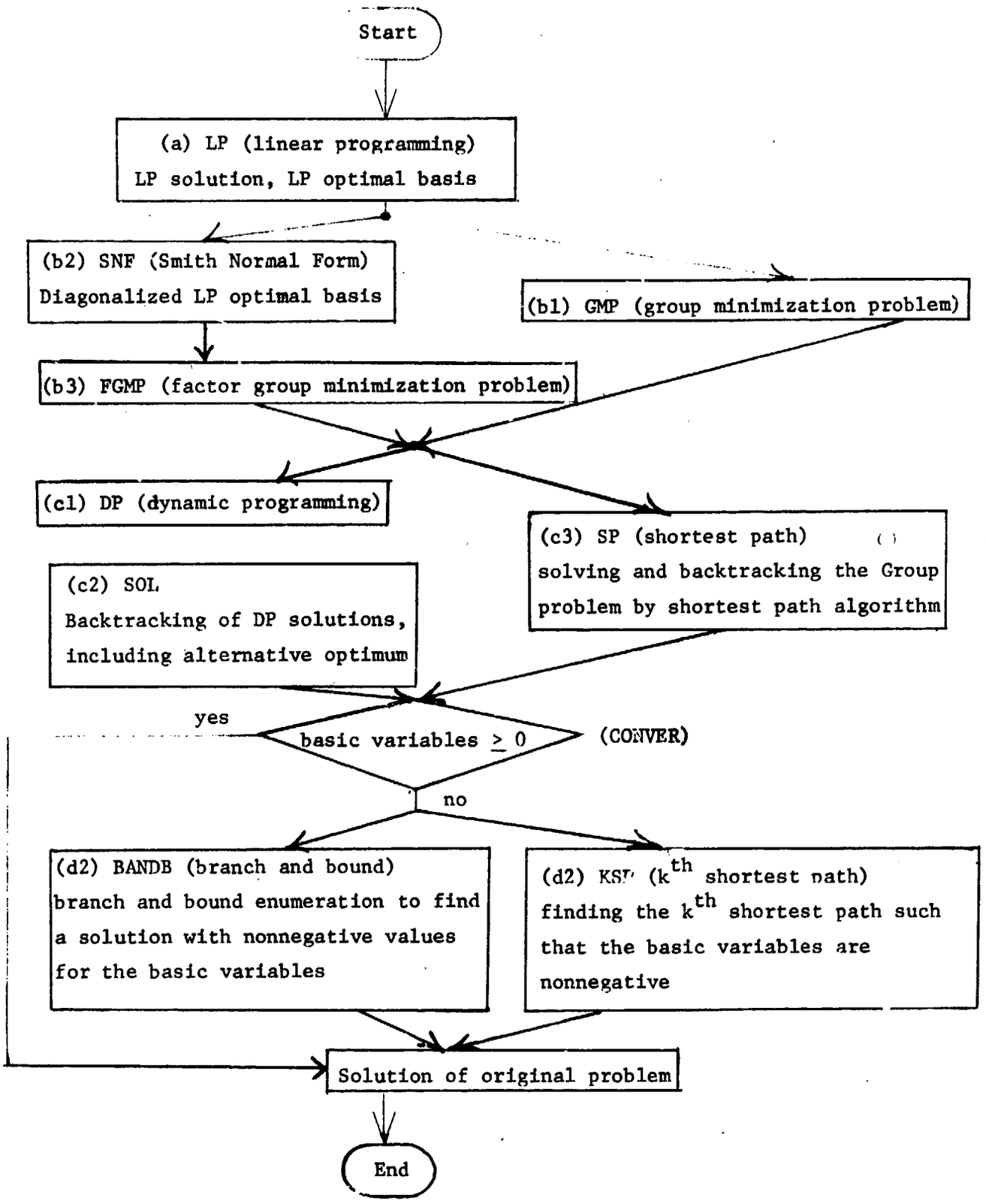

FIGURE 1: THE GENERAL OUTLINE OF THE GROUP THEORETIC ALGORITHM

easier handling allowed in the other procedure. The selection of the group is made by an input parameter.

A flow chart representing the construction of a group problem appears in Figure 2 and that of constructing SNF in Figure 3.

(c) Solving the Group Problem

As noted before, GMP or FGMP can be solved by either dynamic programming or a shortest path algorithm.

(c1) (c2) DP and SOL

The subroutine DP solves the group problem using a dynamic programming algorithm, whereas subroutine SOL keeps track of the indices of the solution to the group problem. The main feature of this part of the program is that it can list as many alternative optimum as computer storage allows. It is desirable to obtain all optimal solutions to each group problem, because only some of them may yield nonnegative values for the basic variables. This is equivalent to recording more than one index, say $j(k,g,\ell)$, $\ell=1, \ldots, \ell_{max}$ when ties occur in computing the minimum in the recursive relationship (2). The current program is capable of storing up to five indices, i.e., $\ell_{max} = 5$, for each element of the group. It can be expanded if computer storage allows. If there is more than ℓ_{max} indices due to ties, a new index will not be stored. In this sense, "all" alternative optimum may not be found. A flow chart representing a general outline of the DP algorithm is in Figure 4.

(c3) SPA

This subroutine obtains the network representation of the group problem, and then uses a shortest path algorithm. Retrieving the values of the solution is the same as (c2). There are several alternatives for this algorithm. Conventional algorithms are given by Dijkstra [D2], Dantzig [D3], Floyd [F1], Farbey et.al.[F2], and Yen [Y1]. Moreover, the resulting network is highly structured,

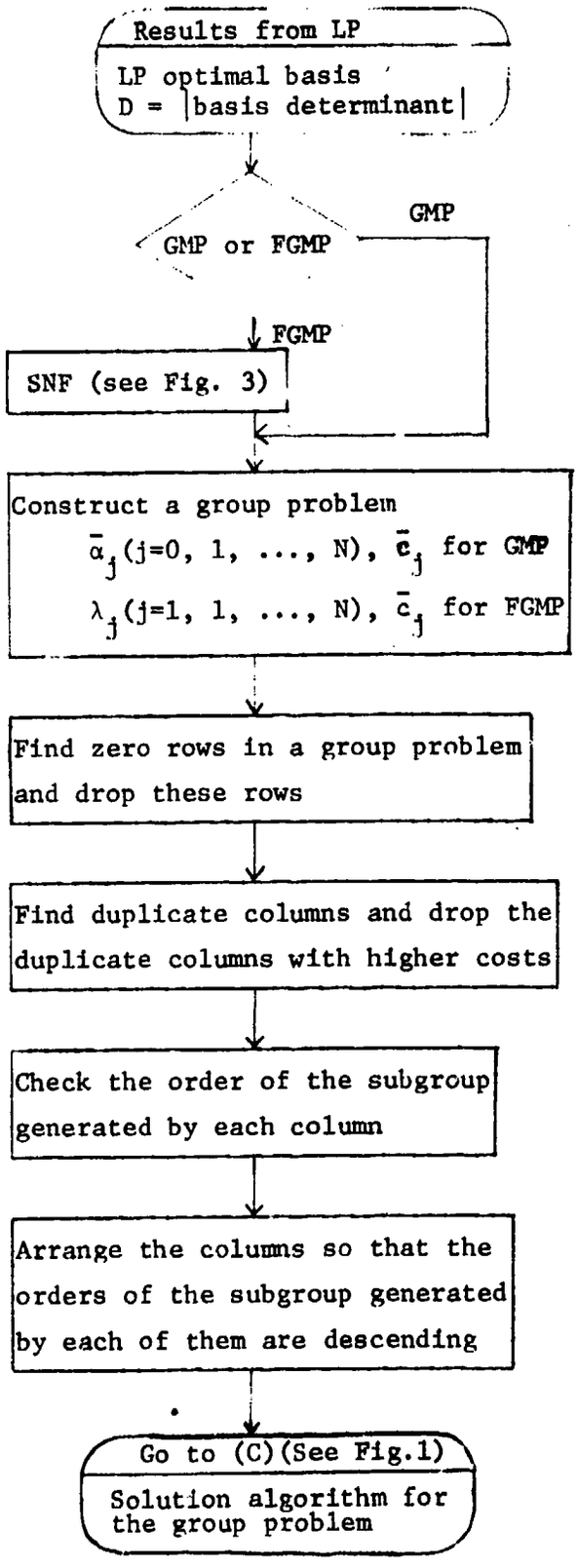

FIGURE 2: GENERAL FLOW DIAGRAM OF THE CONSTRUCTION OF A GROUP PROBLEM

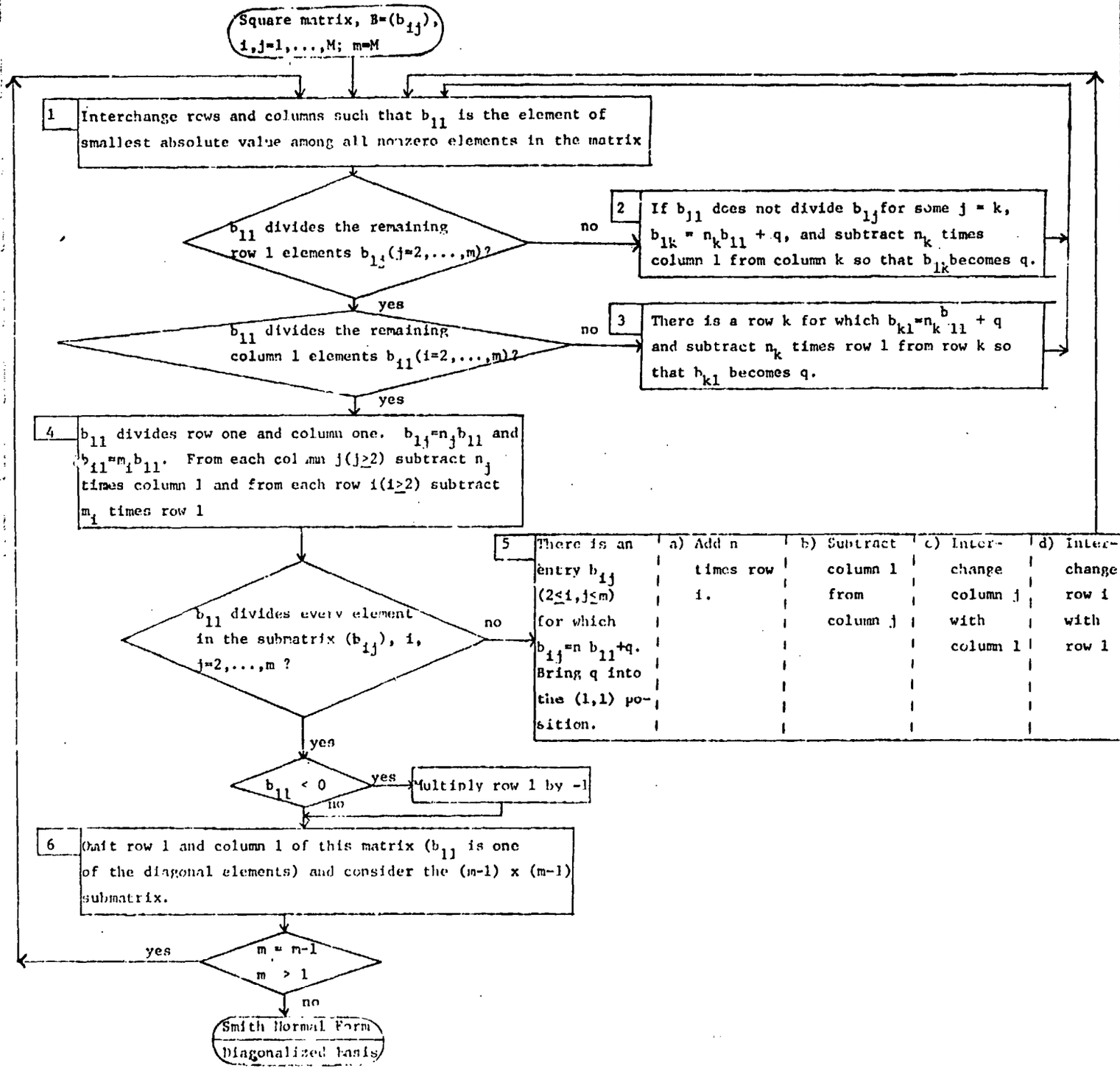

FIGURE 3: FLOW DIAGRAM OF THE SNF COMPUTATION

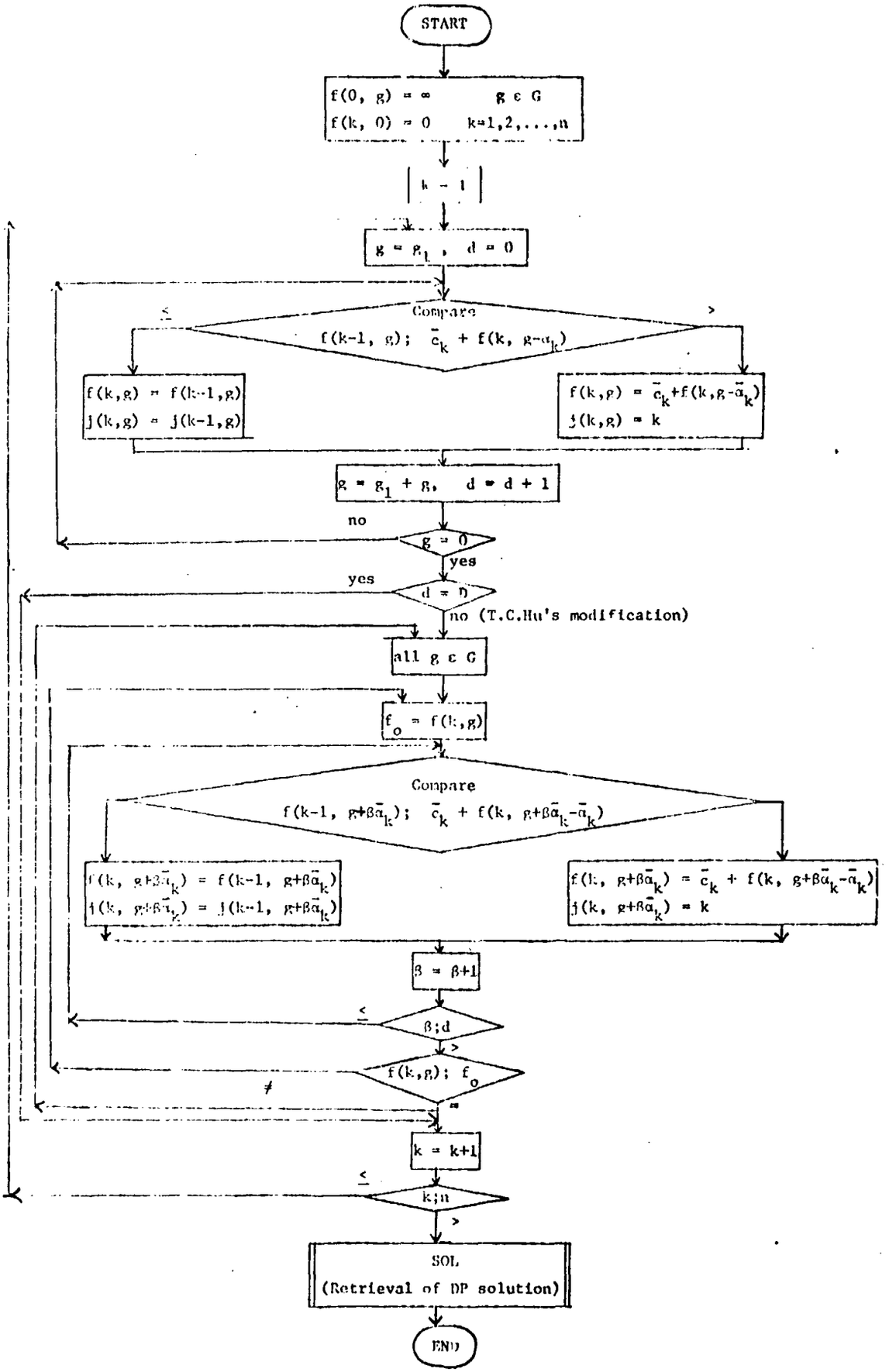

FIGURE 4: A FLOW CHART REPRESENTING THE DP ALGORITHM

e.g., there are as many arcs into a node as there are out of it, and the costs are also symmetric. This suggests a special purpose procedure should be used (see also Chen and Zionts [C1]). Our SPA program is based on Yen's algorithm [Y1] as it appears to require the fewest number of computations.

To examine the structure of the network, consider the example below.

Minimize $3 x_1 + 4 x_2 + 5 x_3$

subject to $\begin{pmatrix}4\\0\\0\end{pmatrix}x_1 + \begin{pmatrix}0\\6\\0\end{pmatrix}x_2 + \begin{pmatrix}0\\0\\6\end{pmatrix}x_3 \equiv \begin{pmatrix}8\\6\\0\end{pmatrix} \pmod{12}$

and $x_1, x_2, x_3 \geq 0$, integer.

The associated D (distance) matrix for this problem appears in Figure 6. Note that the order of $\bar{\alpha}_1$ (denoted NO(1)) = 3, NO(2) = NO(3) = 2, and $G(\bar{\alpha}_j) \neq G(\bar{\alpha})$ = 12 for all j. Then the distance matrix of the network appears as in Figure 5.

Once the distance matrix is constructed, the shortest path algorithm is used. The steps in Yen's shortest path algorithm is described here. We use the following notation:

(I), I = 0, ...,D-1; the nodes of the network (Dnodes) where (0) is the origin.

H(I), I = 0, 1, ...,D-1; the node number stored in the I^{th} cell of H.

(I,J), I ≠ J; the directed arcs from (I) to (J).

d = [d(I,J)]; the distance matrix where d(I,J) is the distance of arc (I,J).

F(I), I = 0, 1, ...,D-1; the tentative or permanent shortest distance from (0) to (I).

Then F(J*)'s, the lengths of shortest paths from (0) to each (J), can be obtained as follows:

I. Let L = 0, K = D-1, F(0) = 0, H(I) = I, and F(I) = ∞ for I = 1, 2, ...,D-1.

II. For I - 1, 2, ...,D-1, do steps A, B, C as follows:

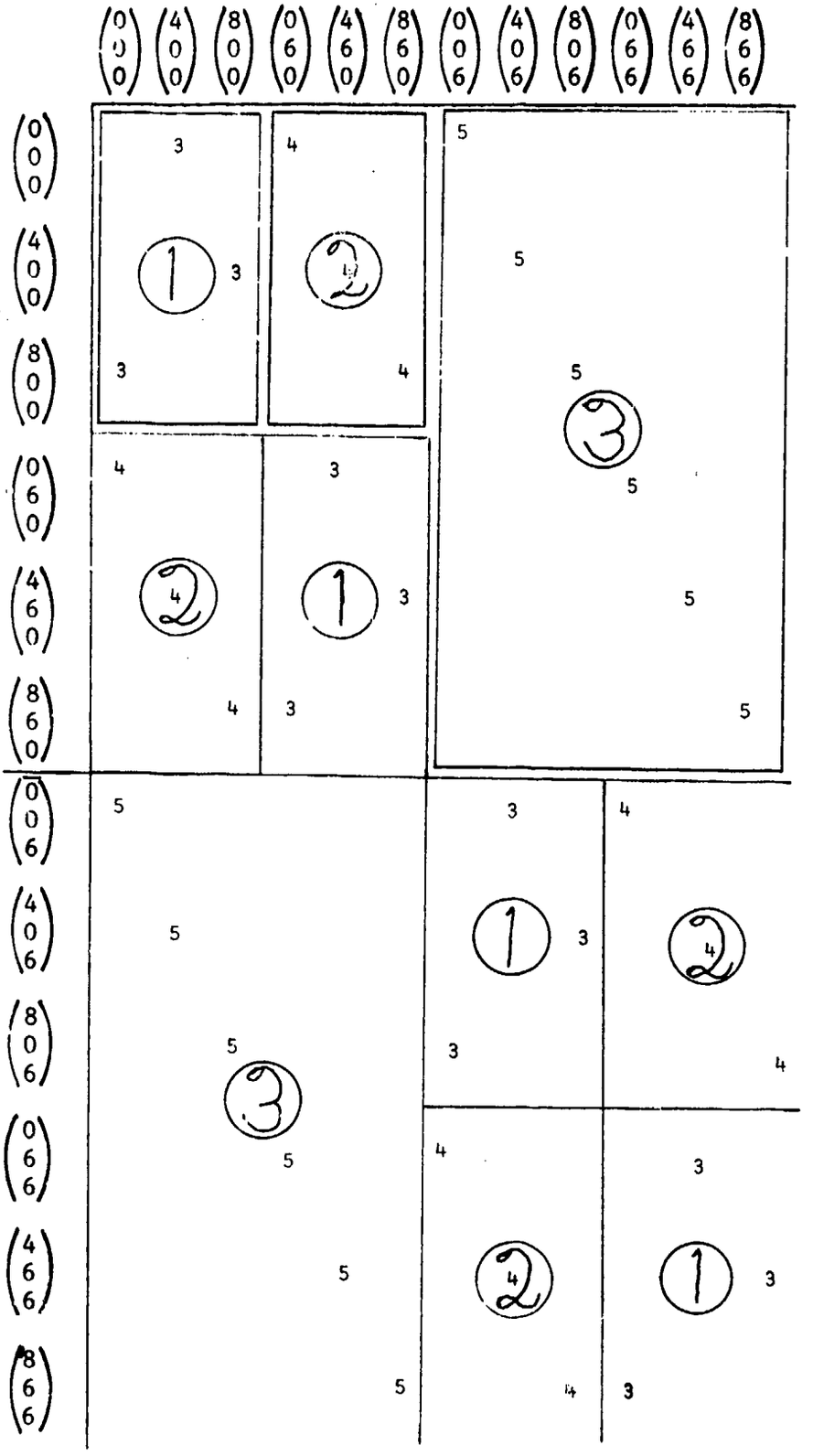

FIGURE 5: D MATRIX

A. Let J = H(I).

B. Compute F(J) = min [F(J) + d(L, J)].

C. If the value of F(J), say F(J*), is less than the current minimum during this execution of step II, replace J* by J and note the corresponding value of I by setting I* = I.

III. Label L = J* and H(I*) = H(K).

IV. Let K = K-1. If K > 0 go to II; otherwise, if K = 0, stop.

(d) The Case When $x_B \not\geq 0$

A subroutine CONVER finds the solution to the integer program from the optimal solution to the group problem. If the optimal solution, x_N, for the relaxed group minimization problem yields nonnegative basic solution, x_B, then (x_B, x_N) is a solution to the original IP problem. If, on the other hand, one or more of the basic variables turn out to be negative, we have to seek the smallest value of $\bar{c}x_N$ for which $x_B \geq 0$ and x_N satisfies the constraints of the group problem. At least three alternatives exist here; namely

(d1) A branch and bound enumeration

(d2) A k^{th} shortest path algorithm

(d3) A modified dynamic programming algorithm.

The current program includes (d1). (d2) is now being incorporated.

(d1) BANDB

A modified dynamic programming algorithm was developed by White [W1] which seeks the smallest shortest route for which $x_B \geq 0$. However, this involves considerable computations. Instead of using this recursive relationship, we will introduce a branch and bound enumeration which is similar to the Dakin [D1] variation. A flow chart of this subroutine is in Figure 6. Our branching strategy is to check <u>all</u> branches emanating from a chosen node. For example, in Figure 7 with 3 nonbasic variables, there exist three possible branches leading to ℓ, ℓ+1, ℓ+2 nodes emanating from node k. All of them are examined.

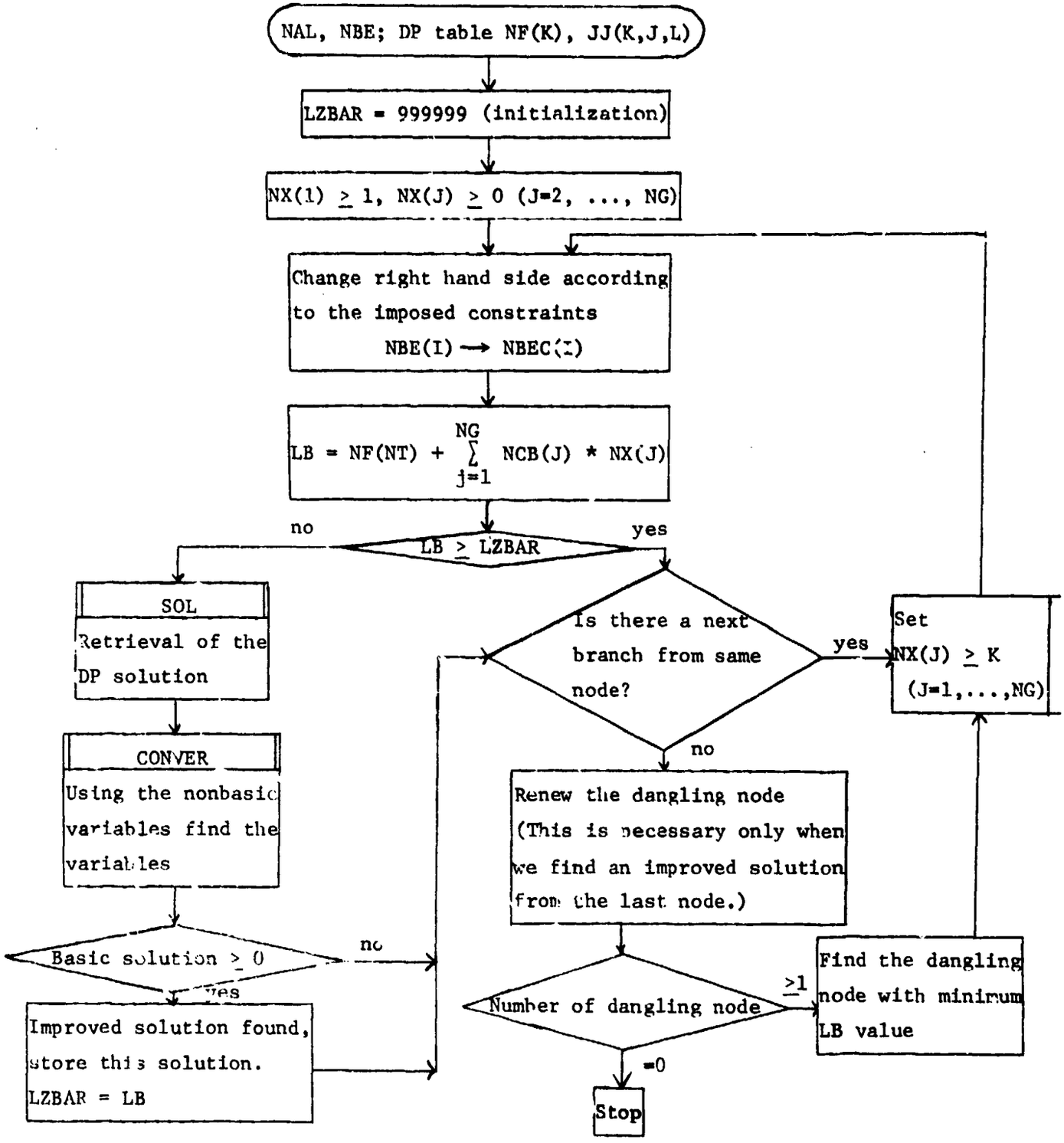

FIGURE 6: A GENERAL OUTLINE OF THE BRANCH AND BOUND ENUMERATION

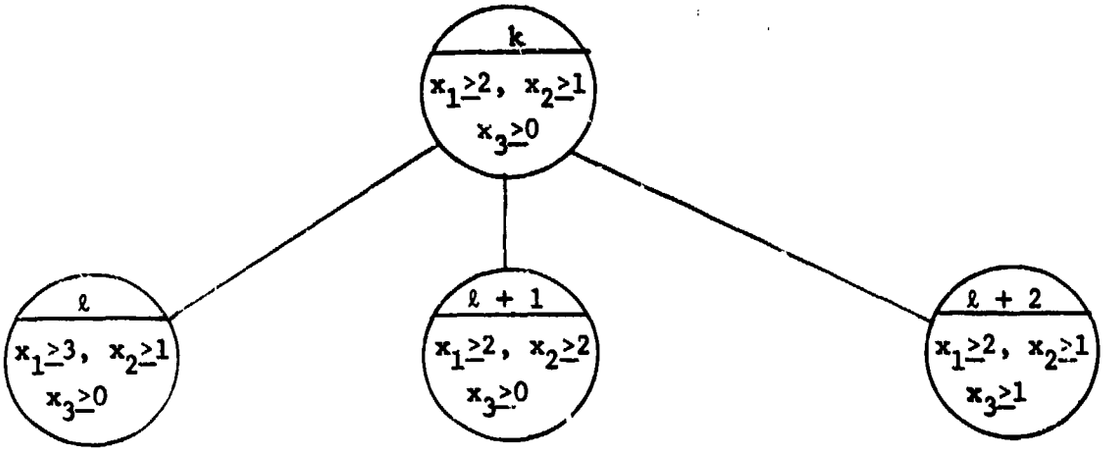

FIGURE 7: BRANCHING

As we impose more strict conditions, the feasible region is more restricted, thus increasing the optimal value of the objective function. The node selection rule is: find the node with the minimum of the optimal linear programming objective values of all dangling nodes. This node is selected as the one from which we branch on. Hopefully, it will produce an optimal solution to the group problem quickly. In the current version of the program as many as 300 dangling nodes can be stored.

3. COMPUTATIONAL EXPERIENCE

The algorithm has been coded in FORTRAN IV and tested on a UNIVAC 1108 computer at Chi Corporation, Cleveland, Ohio. At this time only rather modest sized IBM test problems appearing in Haldi [H2] have been solved. Table 3 gives a detailed summary of running times in seconds for each phase of the system. Unfortunately, some I/O time is included.

Table 3 also includes the results of other computational studies, i.e., White [W1] and Shapiro [S3]. White (1966) used an IBM 7094, which is substantially slower than the results using machines such as a UNIVAC 1108 or IBM 360. Still, it is observed, in general, that the shortest path algorithm works better than the DP algorithm. There is not much difference between Shapiro's

results and our results. Exceptions are those requiring branch and bound enumeration, for which Shapiro's algorithm works better. Also note that for IBM test problems no. 1 and no. 2 with D = 32, our DP technique works faster than Shapiro's algorithm.

4. SUGGESTED STRATEGIES

Table 4 summarizes some general results and inferences suggested by the computational results. From these, we can draw at least preliminary conclusions as to which form of the group problem should be used, GMP or FGMP, and which algorithm should be adopted, DP or Network. Generally speaking, for small problems (i.e., ones with a small number of rows and a small basis determinant in absolute value), GMP works satisfactorily, but for large problems FGMP is definitely preferred. We elaborate upon this and list our suggested algorithm strategies in Table 5. The reader should realize, however, that these conclusions are drawn upon very preliminary computations.

Table 5 indicates that:

1) when both m and D are small, any approach should work.
2) when the group is cyclic, the network approach without the SNF (i.e., use GMP) is best, since we only have to find n_k such that $g_k = n_k g_1$, where g_1 generates the group. The SNF is worse when the number of nonbasic variables, that is, the number of indices k, is small. Thus, we feel GMP/NETWORK is better than FGMP/NETWORK, when the group is cyclic.
3) when the group is not cyclic, the network approach with SNF is usually best, since the SNF can be used to find the number of cyclic subgroups which generates the group. Elements which generate the group can be used to find the entire network.

DP Algorithm and the SPA Algorithm

It will be clear from Table 3 that, generally speaking, SPA works from 5 to 10 times faster than DP. This trend is much clearer as D increases. For

Problem name	Haldi No. 1				Haldi No. 2			
Size	6 x 5				6 x 5			
|det B|	183				258			
Number of cyclic subgroups	1				1			
	DP Algorithm (DP)		Shortest Path Algorithm (SPA)		DP		SPA	
	GMP	FGMP	GMP	FGMP	GMP	FGMP	GMP	FGMP
LP	0.0784	0.0776	0.0800	0.0780	0.0766	0.0772	0.0776	0.0798
GMP (SNF time)	0.0348	0.2122 (0.0894)	0.0576	0.2124 (0.0906)	0.0416	0.2296 (0.0878)	0.0398	0.2296 (0.0890)
Network representation	-	-	0.0218	0.0214	-	-	0.0206	0.0202
DP or SPA	3.6628	3.6890	0.5676	0.5752	7.7366	7.4792	1.0878	1.0176
SOL	1.8014	1.7408			1.2160	1.2004		
CONVER	0.0202	0.0218	0.0060	0.0052	0.0050	0.0052	0.0050	0.0052
BANDB	-	-	-	-	-	-	-	-
TOTAL	5.5986	5.7417	0.7330	0.8922	9.0758	8.9916	1.2308	1.3526
Other computational experience	36.0 White [W1]			1.28 Shapiro [S3]	59.0 White			0.97 Shapiro

TABLE 3(a): RESULTS I

Problem Name	Haldi No. 3				Haldi No. 4			
Size	6 x 5				6 x 5			
|det B|	320				205			
Number of cyclic subgroups	1				1			
	DP		SPA		DP		SPA	
	GMP	FGMP	GMP	FGMP	GMP	FGMP	GMP	FGMP
LP			0.0816	0.0784	0.0782	0.0792	0.0818	0.0802
GMP (SNF time)			0.0498	0.2686 (0.0890)	0.0372	0.2132 (0.0880)	0.0380	0.2252 (0.0930)
Network representation			0.0264	0.0248	-	-	0.0098	0.0090
DP or SPA	(Computation due to [illegible] implementation.)		1.5150	1.5312	4.8865	4.4694	0.7232	0.6940
SOL					0.8048	0.7712		
CONVER			0.0052	0.0052	0.0052	0.0050	0.0054	0.0050
BANDB			-	-	-	-	-	-
TOTAL			1.6780	1.9082	5.8118	5.5380	0.8582	1.0134
Other computational experience	68.0 White			1.27 Shapiro				0.91 Shapiro

TABLE 3(b): RESULTS II

Problem Name	IBM No. 1				IBM No. 2			
Size	7 x 7				7 x 7			
|det B|	32				32			
Number of cyclic subgroups	4				4			
	DP		SPA		DP		SPA	
	GMP	FGMP	GMP	FGMP	GMP	FGMP	GMP	FGMP
LP	0.0814	0.0836			0.0830	0.0928	0.0884	
GMP (SNF time)	0.0220	0.1880 (0.1616)			0.0220	0.2216 (0.1898)	0.0234	
Network representation	-	-		(Not run)	-	-	0.1078	(Not run)
DP or SPA	0.4632	0.4638			0.4458	0.5636	0.0238	
SOL	0.0306	0.0296			0.0220	0.0230	0.0008	
CONVER	0.0232	0.0226			0.0078	0.0082	0.0072	
BANDB	-	-			-	-	-	
TOTAL	0.5204	0.7876			0.5806	0.9092	0.2514	
Other computational experience	12.0 White			2.19 Shapiro	11.0 White			1.95 Shapiro

TABLE 3(c): RESULTS III

Problem Name	Set Covering No. 1				Salkin			
Size	15 x 32				6 x 12			
|det B|	18				114			
Number of cyclic subgroups	1				1			
	DP		SPA		DP		SPA	
	GMP	FGMP	GMP	FGMP	GMP	FGMP	GMP	FGMP
LP	0.3316	0.3298	0.3388	0.3310	0.0972	0.0902	0.0884	0.0878
GMP (SNF time)	0.1446	3.3842 (3.1510)	0.1484	3.3686 (3.1370)	0.0546	0.2340 (0.0984)	0.0472	0.2332 (0.0994)
Network representation	-	-	0.0094	0.0044	-	-	0.0212	0.0192
DP or SPA	0.3178	0.1876	0.0224	0.0212	3.8476	3.3606	0.2066	0.2014
SOL	0.1296	0.1042			0.0670	0.0740		
CONVER	0.0766	0.0770	0.0796	0.0758	0.0086	0.0083	0.0086	0.0086
BANDB	-	-	-	-	-	-	-	-
TOTAL	1.0002	4.0828	0.5986	3.8010	4.0750	3.7573	0.3720	0.5416

TABLE 3(d): RESULTS IV

Problem Name	IBM No. 3				Hefley No. 13		White	
Size	3 x 4				6 x 26		3 x 2	
|det B|	72				48		8	
Number of cyclic subgroups							1	
	DP		SPA		SPA		SPA	DP
	GMP	FGMP	GMP	FGMP	GMP	FGMP	GMP	FGMP
LP	0.0462	0.0428			0.4522		0.0476	0.0472
GMP (SNF time)	0.0134	0.0346 (0.0118)			0.0422		0.0098	0.0210 (0.0094)
Network Representation	-	-			0.2064		0.0008	-
DP or SPA	0.5238	0.7798	(Not run)		0.0342	(Not run)	0.0118	0.0294
SOL	0.0138	0.0258			0.0006			0.0046
CONVER	0.0020	0.0022			0.0128		0.0016	0.0016
BANDB	10.5686	11.5886			-		0.3012	0.3008
TOTAL	11.1728	12.4738			0.7484		0.3728	0.4046
Other computational experience	27.0 White			0.82 Shapiro	12.05 Hefley			

TABLE 3(e): RESULTS V

	GMP	FGMP
Dynamic programming	. No SNF conversion time is required. . This will probably be inefficient especially with D large, because of the number of states (i.e., D). This will require more memory.	. Since the number of rows in FGMP, m_F, is usually of the order of 1 to 5 regardless of the value of m this will be more efficient than the GMP/DP strategy, especially in large (many row problems. . Generally the network algorithm will work better.
Network	. This should work well for small (in number of rows and/or the size D) problems at the cost of more memory requirements for the distance matrix. . After converted into a network, the group of elements are not carried over to the network optimization. . No SNF time is required.	. This is the most popular, and probably most efficient strategy. . This should work well for larger problems. . Both SNF conversion time and network representation time are required.

TABLE 4: SOME GENERAL INFERENCES

example, see the computations with the Holdi No. 2 problem (D=258). The difference in time is directly attributed to the difference of DP and SPA time. Even though the values for D are rather small in the problems treated here, SPA seems to be more advantageous. Moreover, SPA requires substantially less computer memory requirements. This is because SPA does not carry the $\bar{\alpha}_j$ or λ_j columns during the network optimizations, since these coefficient columns correspond to group elements and, hence, nodes (see Salkin [S1]).

GMP and FGMP

For the rather small problems solved, the results indicate that diagonalization of the (LP optimal) basis via Smith's Normal Form is not preferable. This may be partially because our computer subroutine representing the SNF conversion is not written well. But, on the other hand, it is intuitively clear that for small (especially in the number of row) problems, diagonalization does not speed up computation time. This will not be the case for larger (in

m \ D	Small	Large
Small	1. GMP/NETWORK** 2. GMP/DP Group cyclic or not cyclic	1. FGMP/NETWORK Group cyclic or not cyclic
Large	1. GMP/NETWORK Group cyclic 1. FGMP/NETWORK Group not cyclic 2. GMP/DP Group cyclic	1. GMP/NETWORK Group cyclic 2. FGMP/NETWORK Group not cyclic

*1. first choice, 2. second choice
**GMP/NETWORK stands for solving GMP by the network approach. Others should be interpreted similarly.

TABLE 5: SUGGESTED STRATEGIES*

terms of rows and the value of D) problems. Further investigation will be needed to conclude under what conditions which is better.

5. FUTURE WORK

Detailed comparisons of our computational results with others indicates that further improvement of the program will be possible in the following subroutines. These minor changes should make the system more efficient.

1) Smith Normal Form

The current subroutine used to obtain the Smith Normal Form is based on the work in Hu [H1]. This subroutine is in a preliminary form and should be refined. Another approach, given by D.A. Smith [S6], is being examined.

2) Branch and Bound Enumeration

Our current subroutine containing the branch and bound enumeration is rather primitive. This is especially true for both node and branch selection rules. This subroutine will become very inefficient as D increases. Modifications are being incorporated in both node and branch selection.

Finally, major points to be studied further are:

(1) a further investigation of the degree of relaxation used to define the GMP.

(2) inclusion of the use of Lagrangian multipliers (as suggested by A. Geoffrion [G8], et al.) and Gomory all integer cuts (as suggested by G.A. Gorry [G5], et al.) in the relaxation strategies,

(3) inclusion of the cuts derived from functions related to corner polyhedra (as suggested by R. Gomory and E. Johnson [G6, G7]) in the implicit enumeration tests,

(4) inclusion of the k^{th} shortest path algorithm (as suggested by B. Fox [F3]) if a group problem does not solve the integer program.

(5) the use of non-optimal dual LP solutions to produce a group problem.

Points (1), (2), and (3) are mentioned by Salkin [S1]. To explain (4) and (5) in more detail: The k^{th} shortest path algorithm is being incorporated into the program. White [W1] shows that the integer program can be solved by finding the k^{th} shortest path, possibly containing loops, so that a non-negative solution to the integer program is produced. Fox [F3], however, points out that White's approach does not perturb the arc lengths, leaving open the possibility of cycling. Another problem is zero and duplicate columns, resulting from reducing each updated coefficient column to its fractional part. As described elsewhere [S5], the k^{th} shortest path of a network associated with a group problem generally does not solve the original integer program when duplicate and/or zero columns are dropped when constructing the group problem. Therefore, a more complicated procedure is required. Our procedure repeatedly uses the k^{th} shortest path as in [S5]. Computational studies are being made to test this procedure.

Another interesting thought is the use of non-optimal dual LP solutions to produce a group problem. In order to generate a group problem, the cost row

$\bar{c} = c_B B^{-1} N - c_N$ must be nonnegative. If we use the dual simplex method, this condition is always satisfied. Thus, we can use any of the dual feasible solutions. It seems natural that a dual feasible solution which is close to a dual optimal (hence, primal optimal) solution is preferred. This suggests using a dual feasible, not necessarily optimal, solution to construct a group problem so that the dual feasible solution

(i) corresponds to a basis whose determinant is small

(ii) yields a cyclic group.

Criterion (i) will make the group problem small, and (ii) will allow a simpler application of the DP or SPA algorithm. Both points allow for a faster solution of the GMP. Computational studies are in progress to test this strategy. A later memorandum will report on these efforts.

6. REFERENCES

[B1] Bellman, R., and Kalaba, R. "On k^{tn} Best Policies." Journal of Soc. Indust. Appl Appl. Math., VIII(4), 1960.

[C1] Chen, D.S., and Zionts, S. "On the Use of Shortest Route Solution Methods to Solve the Group Theoretic Integer Programming Problem." Working Paper Series, No. 140, School of Management, State University of New York at Buffalo, April, 1972.

[D1] Dakin, R. " A Tree Search Algorithm for Mixed Integer-Programming Problems." Computer Journal, VIII(1965), 250-255.

[D2] Dijkstra, E.W. "A Note on Two Problems in Connexion with Graphs." Numberische Mathematik, I (1965), 269-271.

[D3] Dantzig, G.B. Linear Programming and Extensions. Princeton, N.J.: Princeton University Press, 1962.

[F1] Floyd, R.W. "Algorithm 97: Shortest Path." Communications of ACM, V(6), 1962.

[F2] Farbey, F.A.; Land, A.H.; and Murchland, J.D. "The Cascade Algorithm for Finding All Shortest Distances in a Directed Graph." Management Science, XIV(1), September, 1967.

[F3] Fox, B.L. "Calculating k^{th} Shortest Paths." INFOR, IX(1), February 1973.

[G1] Gomory, R.E. "An Algorithm for Integer Solutions to Linear Programs." in Recent Advances in Mathematical Programming. Edited by R. Graves and P. Wolfe. New York: McGraw Hill, 1963.

[G2] Gomory, R.E. "On the Relation between Integer and Noninteger Solutions to Linear Programs." Proc. Nat. Acad. Sci., 53.

[G3] Gorry, G.A.; Northrup, W.D.; and Shapiro, J.F. "Computational Experience with a Group Theoretic Integer Programming Algorithm." Mathematical Programming, IV(2), April 1973.

[G4] Glover, F. "Integer Programming over a Finite Additive Group." SIAM Journal on Control, VII(2), May, 1969.

[G5] Gorry, G.A.; Shapiro, J.F.; and Wolsey, L. "Relaxation Methods for Pure and Mixed Integer Programming Problems." Management Science, XVIII(5), 1972.

[G6] Gomory, R., and Johnson, E. "Some Continuous Functions Related to Corner Polyhedra." Mathematical Programming, III(1) (August, 1972), 23-86.

[G7] "Some Continuous Functions Related to Corner Polyhedra, II." Mathematical Programming, III(3) (December, 1972), 359-389.

[G8] Geoffrion, A.M. "Lagrangean Relaxation for Integer Programming." Working Paper No. 195, Western Management Science Institute, University of California, Los Angeles, December 1972 (Revised December, 1973).

[H1] Hu, T.C. Integer Programming and Network Flows. Reading, Mass.: Addison-Wesley Publishing Co., 1969

[H2] Haldi, J. "25 Integer Programming Test Problems." Working Paper No. 43, Graduate School of Business, Stanford University, December, 1964.

[H3] Hoffman, W., and Parley, R. "A Method for the Solution of the N^{th} Best Path Problem." Journal of ACM, VI(4), 1959.

[H4] Hefley, G., and Thomas, M. "Decomposition of the Group Problem in Integer Programming." Report No. IME 72-101, Department of Industrial Management Engineering, University of Iowa at Iowa City, 1972.

[S1] Salkin, H.M. Integer Programming. Reading, Mass.: Addison-Wesley Publishing Co., January, 1975.

[S2] Salkin, H.M., and Spielberg, K. "Set Covering Algorithm One (SCA 1)." IBM New York Scientific Center, August, 1969.

[S3] Shapiro, J.F. "Group Theoretic Algorithms for the Integer Programming Problem - II: Extension to a General Algorithm." Operations Research, XVI(5), 1968.

[S4] Sakarovitch, M. "The k Shortest Routes and the k Shortest Chains in a Graph, Operations Research Center, University of California at Berkeley, Report ORC-32, October, 1966.

[S5] Salkin, H.M., and Morito, S. "A Note on the Shortest Path Algorithm in the Group Minimization for Integer Programming." Memo, Department of Operations Research, Case Western Reserve University, forthcoming.

[W1] White, W.W. "On a Group Theoretic Approach to Linear Integer Programming." ORC 66-27, Operations Research Center, University of California at Berkeley,, 1965.

[W2] Williams, T.A., and White, G.P. "A Note on Yen's Algorithm for Finding the Length of All Shortest Paths in N-Node Nonnegative-Distance Networks." Journal of ACM, XX(3), July, 1972.

[Y1] Yen, J.Y. "Finding the Lengths of All Shortest Paths in N-Node Non-negative Distance Complete Network Using $1/2\ N^3$ Additions and N^3 Comparisons." Journal of ACM, XVIV(3), July, 1972.

[Y2] "Finding the k Shortest Loopless Paths in a Network." Management Science XVII(11), July, 1971.

APPENDIX I: NOTATIONS

1) Constants

Program variable name	Usual Notations	
M	m	Number of rows } of the original problem
N	n	Number of columns } of the original problem
MM		Number of rows } of the group problem
NG		Number of columns } of the group problem
ND	$\|\det B\|$	Absolute value of the determinant of basis B.
NDD (=NO(J))		The order of the cyclic subgroup generated by the j^{th} column of the group problem.

2) Arrays

NA(J,I)	a_{ij}	Coefficient matrix } of the original problem
NB(I)	b_i	Right hand side } of the original problem
NC(J)	c_j	Costs
NAB(J,I)		Basis of LP optimal solution.
NB1(J,I)		Basis inverse (fractions are cleared by multiplying by ND = $\|\det B\|$).
NXB(J)		Indices of basic variables
NAL(J,I)	$\bar{\alpha}_{ij}$	Coefficient matrix } of the group problem
NBE(I)	$\beta_i = a_{0i}$	Right hand side } of the group problem
NCB(J)	$\bar{c}_j$	Costs
NO(J) (=NDD)		The order of the cyclic subgroup generated by the j^{th} column of the group problem.
MB(I)	ε_i	ε_i generated by the SNF.
NXSOL		Solution for the group problem.
NF(K,NT)	$f(k,g)$	DP table } (K=1,2)
JJ(K,NT,L)	$j(k,g,\ell)$	DP table } (K=1,2)

3) Working Indices

NT = 1, ND (or NDD)	For group elements.
K or NGG or J	For columns of the group problem.
I	For rows of the group or original problem.

APPENDIX II: SOME COMMENTS ON THE k^{th} SHORTEST PATH ALGORITHM

As mentioned earlier, an alternate to the enumeration is an algorithm which seeks the smallest shortest route for which $x_B \geq 0$. A k^{th} shortest path algorithm may be used. There are two types of k^{th} shortest paths problems. The first is to find the k^{th} shortest paths from the origin to the sink in which loops are allowed, the second, in which loops are not allowed. We must use the first type, since the smallest shortest route for which $x_B \geq 0$ may contain loops [S1]. Fox [F3] developed a k^{th} shortest path algorithm which can be applied to the group problem, but no computational comparison among alternative strategies is available. Other k^{th} shortest path algorithms with loops are in Hoffman and Parley [H3] (1959), Bellman and Kalaba [B1] (1960), and Sakarovitch [S4] (1966).