

DUDLEY KNOX LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIF 93940

NAVAL POSTGRADUATE SCHOOL

Monterey, California



THESIS

AUTOMATIC IDENTIFICATION OF NETWORK ROWS
IN LARGE-SCALE OPTIMIZATION MODELS

by

William Gordon Wright

September 1980

Thesis Advisor:

G. G. Brown

Approved for public release; distribution unlimited.

T196278

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Automatic Identification of Network Rows in Large-Scale Optimization Models		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis; September 1980
7. AUTHOR(s) William Gordon Wright		6. PERFORMING ORG. REPORT NUMBER
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		8. CONTRACT OR GRANT NUMBER(s)
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		12. REPORT DATE September 1980
		13. NUMBER OF PAGES 51
		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution 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) Networks Large-Scale Optimization Basis Factorization Computational Complexity Mixed Integer Optimization Generalized Upper Bounds		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) The solution of contemporary large-scale linear, integer, and mixed integer programming problems is often facilitated by the exploitation of intrinsic special structure in the model. This paper deals with the problem of identifying embedded pure network rows within the coefficient matrix of such models and presents two heuristic algorithms for identifying such structure. The problem of identifying the maximum size embedded pure network is shown to be among the class of NP-hard problems, therefore, the		

#20 - ABSTRACT - CONTINUED

polynomially-bounded algorithms presented here do not guarantee network sets of maximum size. However, upper bounds on the size of the maximum network set are developed and used to evaluate the algorithms. Finally, the algorithms were tested with a number of large-scale, real-world models and the results of these benchmark runs are presented.

Approved for public release; distribution unlimited.

Automatic Identification of Network Rows
in Large-Scale Optimization Models

by

William Gordon Wright
Captain, United States Marine Corps
B.A., Washington State University, 1973
B.S., Washington State University, 1973

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL

September 1980

ABSTRACT

The solution of contemporary large-scale linear, integer, and mixed integer programming problems is often facilitated by the exploitation of intrinsic special structure in the model. This paper deals with the problem of identifying embedded pure network rows within the coefficient matrix of such models and presents two heuristic algorithms for identifying such structure. The problem of identifying the maximum size embedded pure network is shown to be among the class of NP-hard problems, therefore, the polynomially-bounded algorithms presented here do not guarantee network sets of maximum size. However, upper bounds on the size of the maximum network set are developed and used to evaluate the algorithms. Finally, the algorithms were tested with a number of large-scale, real-world models and the results of these benchmark runs are presented.

TABLE OF CONTENTS

I.	INTRODUCTION -----	6
II.	PROBLEM DEFINITION AND REPRESENTATIONS -----	11
	A. THE LINEAR PROGRAMMING PROBLEM -----	11
	B. THE GUB PROBLEM -----	11
	C. REPRESENTATIONS OF THE GUB PROBLEM -----	12
	D. THE NET PROBLEM -----	14
	E. REPRESENTATIONS OF THE NET PROBLEM -----	15
	1. Double-GUB Set Representation -----	16
	2. Vector Space Representation -----	17
III.	IMPLEMENTATION OF AUTOMATIC NETWORK IDENTIFICATION HEURISTICS -----	20
	A. DOUBLE-GUB (D-GUB) ALGORITHM -----	20
	B. NETWORK IDENTIFICATION ALGORITHM -----	22
IV.	PROBLEM COMPLEXITY -----	29
V.	UPPER BOUNDS ON MAXIMUM NETWORK SET SIZE -----	35
VI.	COMPUTATIONAL RESULTS -----	40
VII.	EXTENSIONS -----	42
VIII.	CONCLUSIONS -----	44
	APPENDIX A -----	45
	LIST OF REFERENCES -----	49
	INITIAL DISTRIBUTION LIST -----	51

I. INTRODUCTION

The success of mathematical optimization in Operations Research and the increase in size and speed of digital computers have led to the formulation of very large and complex systems as mathematical programming models. The direct solution of the associated linear programming (LP) problems using the classical simplex method is often prohibitively expensive, if not impossible in a practical sense.

The large-scale models are predominantly characterized by sparse coefficient matrices and inherent special structure. If special structure can be identified, it can often be used to reduce the problem to a more manageable size. Methods so used to exploit special structure can be classified as indirect methods or direct methods.

Indirect methods use special structure to decompose a large problem into smaller related problems. The solution of these sub-problems is then used to construct a solution to the larger problem. Indirect methods are not considered in this paper.

Direct methods attempt to solve the original problem more efficiently by using specializations of the simplex procedure which exploit the special structure. These specializations often replace arithmetic operations with logical operations and use information concerning the nature of the eventual

solution provided by the special structure to reduce computational effort.

This paper is concerned with structures which can be used in direct, static factorization algorithms, for which all simplex bases share a common structure under row partition. The details of actual exploitation of special structure, once identified, will not be discussed here (see Graves and McBride [6]).

Static basis factorizations include simple upper bounds, generalized upper bounds (GUB), and embedded network rows, among others. Simple upper bound rows have only one non-zero coefficient. GUB refers to a set of rows for which each column (restricted to those rows) has at most one non-zero coefficient. Embedded network rows refers to a set of rows for which each column (restricted to those rows) has at most two non-zero coefficients of opposite sign. If the non-zero coefficients in the embedded network rows are restricted to one +1 and one -1 in each column, then the structure is referred to as an embedded pure network (NET).

Various transformations are available to identify and exhibit special structure in the coefficient matrix. These range from simple permutation of rows and columns to full (linear) transformations of the coefficient matrix. An intermediate method allows simple scaling (multiplication by a non-zero constant) of each row and/or column. Generally, complete transformation methods are used in an attempt to

convert the entire coefficient matrix to one having a very special structure, such as a node-arc incidence matrix for a network. Partial transformation methods look for large subsets of rows in the coefficient matrix which exhibit the desired structure, with the implicit presumption that large subsets are more efficiently exploited than small subsets.

As mentioned earlier, much of the computational improvement of the specialized simplex algorithms is obtained when logic can be substituted for arithmetic in simplex operations. This is most conveniently accomplished when the coefficient values in the rows of the special structure set are restricted to $0, \pm 1$. This restriction can be satisfied by considering for inclusion in the structure set only those rows with intrinsic $0, \pm 1$ entries. In practice, however, it is often possible, through row and/or column scaling, to produce the desired $0, \pm 1$ values. For simple upper bounds, row scaling will suffice. GUB sets can be converted with row and column scaling (except that columns corresponding to integer variables are not customarily scaled). To produce pure network rows, however, the scaling problem is non-trivial due to the existence of two non-zero coefficients in many columns as well as the requirement that unit elements in the same column be of opposite sign.

The use of generalized upper bounds has received much attention since the concept was introduced in 1964 by Dantzig and Van Slyke [4]. Some form of GUB has been implemented in

many commercial LP systems, though restrictions on what constitutes an admissible (i.e., implemented) GUB set vary. Work has been done in the automatic identification of GUB sets [2], [8]; computational results on large-scale problems indicate that this is not only feasible, but can be extremely advantageous [3], [13].

This paper extends the research done in identifying GUB sets to finding sets of embedded pure network rows. In pure network rows, coefficients are restricted to the values 0, +1 and columns with two non-zero entries must have exactly one +1 entry and one -1 entry. Although some recent work has been done in the theory of complete conversion of a linear program to a network problem ([1], [10]), few practical results have been achieved which reliably identify a subset of rows which form a network structure if complete conversion fails. An efficient algorithm for doing so is of great value since the problem usually fails to be completely convertible, and since the expense of attempting complete conversion may be prohibitive.

The problem of finding a maximum GUB set (in terms of number of rows) within a general coefficient matrix has been shown by Thomen to be NP-hard in complexity [13]. This paper proves the same result for the maximum embedded pure network problem. The implication is that currently only exponential time algorithms exist to solve these types of problems and the hope of finding a more efficient algorithm is dim.

Therefore, the efficient GUB-finding methods developed to date have been heuristic algorithms. That is, they use methods which attempt to find large, sometimes even maximum GUB sets, but which cannot guarantee a maximum GUB set. Since the size of the maximum GUB set is not known for the large-scale problems with which we work, we must resort to comparison with upper bounds on the size of the maximum set to evaluate the heuristics [13].

The methods developed in this paper for finding pure network row sets are also heuristic algorithms. Bounds are developed for the maximum number of pure network rows, in order to evaluate the NET identification algorithms.

Computational results are given for a number of large-scale, real-world problems. They show the NET identification algorithms to be very efficient and effective in identifying large sets of pure network rows.

Portions of this research have been published in abstract form in [14].

II. PROBLEM DEFINITION AND REPRESENTATIONS

A. THE LINEAR PROGRAMMING PROBLEM

The Linear Programming Problem is defined here as:

$$\begin{aligned} \text{(L)} \quad & \text{minimize } c^t x \\ \text{s.t.} \quad & \underline{r} \leq Ax \leq \bar{r} \quad (\text{ranged constraints}) \\ & \underline{b} \leq x \leq \bar{b} \quad (\text{simple bounds}) \end{aligned}$$

where \underline{r} and \bar{r} are m -vectors, x , c , \underline{b} and \bar{b} are n -vectors and A is an $m \times n$ matrix. Consider for the moment the case where all x variables are real-valued; the integer and mixed integer cases will be discussed later.

B. THE GUB PROBLEM

The (maximum) GUB problem for (L) can be stated as:

(GUB) Find a (maximum) subset of rows in A which can be scaled to contain only 0,+1 entries and which satisfy the property that each column of A (restricted to those rows) has at most one non-zero entry.

The real values of the non-zero coefficients in A do not make a difference in the GUB problem, since any non-zero entry

in a GUB row can be scaled to +1 by column scaling alone. Therefore, it is convenient to replace A by a binary (0,1) matrix, K, of the same dimension where each non-zero entry of A is replaced by +1 with all other entries zero.

Using the matrix K, with entries k_{ij} , the (maximum) GUB problem can be formulated as the binary integer program

$$\begin{aligned}
 \text{(GUBI)} \quad & \text{(maximize)} \quad z_1 + z_2 + \dots + z_m \\
 & \text{s.t.} \quad \sum_i k_{ij} z_i \leq 1; \quad j = 1, \dots, n \\
 & \text{where} \quad z_i \in \{0,1\}
 \end{aligned}$$

The variable z_i is an indicator variable for GUB inclusion, i.e., if $z_i = 1$, row i is included in the GUB set, otherwise it is not.

C. REPRESENTATIONS OF THE GUB PROBLEM

Alternative representations of the GUB problem have been developed as the basis for various heuristic algorithms and for theoretical considerations such as determining the complexity of the problem and developing bounds on the maximum achievable size of a GUB set. Some of these representations, covered in detail in [13], are the graphical conflict representation, the conflict matrix representation, and the vector space representation.

The concept of row conflicts is basic to the GUB problem. Two rows in A are said to *conflict* if there is at least one column of A with non-zero entries in both rows. If each row of A is considered as a vertex in an undirected graph with two vertices connected by an edge whenever the corresponding rows conflict, then the (maximum) GUB problem becomes one of finding a (maximum) independent set of vertices in the graph. An independent set of vertices in a graph is a subset of the total vertex set with no two vertices adjacent (connected by an edge) in the graph.

The conflict matrix representation of the GUB problem uses an $m \times m$ symmetric binary matrix M with each row and column representing a row of A . M has +1 values in those i, j entries where row i and row j conflict in A . By definition, every row conflicts with itself so the main diagonal of M has all +1 entries. The (maximum) GUB problem then becomes one of finding (through permutation of the rows of A) an embedded identity matrix (of maximum size) in the conflict matrix M .

The vector space representation has developed from a paper by Senju and Toyoda [12] dealing with the heuristic solution of certain 0,1 programming problems. Each row of K is considered as a vector in n -space having unit length in those directions corresponding to its non-zero entries. The vector R is formed as the sum of each of the row vectors. A unit hypercube in n -space situated at the origin with length 1 in

all positive directions represents the feasible GUB region. If R extends beyond this region, the set of rows is not a GUB set and at least one row must be removed to bring R into the feasible region. The (maximum) GUB problem becomes one of determining (the minimum number of) rows which must be removed in order to bring R into the feasible region. The heuristics based on this representation compute gradient vectors which indicate the direction of shortest distance to the feasible region and remove first those rows which produce the greatest movement in that direction. These heuristics have proven to be the fastest to date and produce GUB sets comparable in quality to other heuristics [13].

D. THE NET PROBLEM

The (maximum) Embedded Pure Network problem for (L) can be stated as:

(NET) Find a (maximum) subset of rows in A which can be scaled to contain only $0, \pm 1$ entries and which satisfy the property that each column of A (restricted to those rows) has at most two non-zero entries, and if the column has two non-zero entries, the (scaled) entries must be of opposite sign.

The real values of the non-zero coefficients in A cannot be ignored as they were in the GUB problem since simple column

scaling is no longer sufficient to produce the required ± 1 entries in columns containing two non-zero entries. The addition of row scaling may help, but even this is not sufficient to guarantee that a network set of rows obtained by considering only the signs of the non-zero elements can be scaled to the required $0, \pm 1$ values.

Considering, for the moment only, matrices with $0, \pm 1$ entries (or a subset of m rows with $0, \pm 1$ entries in a general matrix) with no scaling allowed, the (maximum) NET problem can be formulated as the binary integer program:

$$\begin{aligned}
 \text{(NETI)} \quad & \text{maximize} \quad z_1 + z_2 + \dots + z_m \\
 \text{s.t.} \quad & \sum_{i:a_{ij}=-1} z_i \leq 1; \quad j = 1, \dots, n \\
 & \sum_{i:a_{ij}=+1} z_i \leq 1; \quad j = 1, \dots, n \\
 & \text{where} \quad z_i \in \{0, 1\} \quad .
 \end{aligned}$$

Again, z_i is an indicator variable for inclusion in the network set.

E. REPRESENTATIONS OF THE NET PROBLEM

Unfortunately, NET does not lend itself to the many representations which GUB admits. The primary reason for this is that the scaling problems associated with NET make it impossible to disregard the real values of the non-zero coefficients

in A. Also, the concept of pairwise row conflicts so useful in the GUB algorithms does not apply to network rows when row scaling is allowed.

To efficiently confront the scaling dilemma, we are currently forced to restrict the eligibility of rows for membership in the network set. The most obvious restriction is to allow no scaling and consider only those rows with intrinsic $0, \pm 1$ entries. This may be unnecessarily prohibitive. Two less restrictive options are employed in the algorithms described later. These are:

1. Admit only rows with intrinsic $0, \pm 1$ entries but allow row *reflection* (multiplication of a row by -1).
2. Admit only rows whose non-zero entries can be row scaled to $0, \pm 1$. This includes rows with all non-zero entries of the same absolute value. Row reflection is also allowed.

For the algorithms presented in this paper, two representations of the NET problem are developed. The double-GUB representation is used in the identification of networks with a bipartite structure by considering the network set as consisting of two disjoint GUB sets. The vector space representation is used in the identification of general network structure.

1. Double-GUB Set Representation

As suggested by Thomen [13], GUB heuristics can be used to produce a bipartite network row factorization. This

type of factorization has the property that the rows in the network set can be partitioned into two subsets, N_1 and N_2 , such that each column of the matrix has at most one non-zero entry in N_1 and at most one non-zero entry in N_2 . Additionally, the entries must be of opposite sign. To produce such a factorization, a GUB heuristic can be applied to the eligible rows of A producing one GUB set, and then to the remaining eligible rows not selected in the first pass, producing a second GUB set. This can be done if inclusion in the second GUB set is conditioned on agreement with rows in the first GUB set, allowing row reflection if necessary. This algorithm (entitled D-GUB) was implemented using the fastest of the GUB heuristics.

2. Vector Space Representation

If we consider only the rows of A with $0, \pm 1$ entries, or those which have been scaled to $0, \pm 1$, a vector space representation for NET can be developed similar to that developed for GUB. The representation can also allow reflection of rows, if desired.

With each row in the eligible set, we associate two vectors in n-space, V_i^+ and V_i^- . V_i^+ is the vector consisting of 1 in those dimensions corresponding to +1 entries in row i and zero in all other dimensions. V_i^- is the vector consisting of -1 in those dimensions corresponding to -1 entries in row i and zero in all others. For example, if row i is (1,0,-1,1,0,-1,0), then $V_i^+ = (1,0,0,1,0,0,0)$ and $V_i^- = (0,0,-1,0,0,-1,0)$.

We define R^+ as the resultant vector from the sum of all V_i^+ and R^- as the resultant vector from the sum of all V_i^- . These vectors extend from the origin into the orthants of n-space corresponding to all positive dimensions and all negative dimensions, respectively. Two hypercubes in this n-space, one situated at the origin with length 1 in all positive directions and another situated at the origin with length 1 in all negative directions, constitute the feasible NET region. Should either R^+ or R^- extend beyond its feasible region then the rows in the eligible set do not currently form an admissible set of network rows.

The reflection (multiplication by -1) of a row merely results in the switching of the V^+ and V^- vectors for the row. That is, when row i is reflected, the negative of V_i^+ becomes V_i^- and the negative of V_i^- becomes V_i^+ . This in turn will change the vectors R^+ and R^- . In fact, it is possible in some cases that just the reflection of rows in an infeasible set may bring R^+ and R^- into their feasible regions without deletion of any rows.

This representation is used as the basis for the network identification heuristic for (NET) developed by the author and presented herein. If either R^+ or R^- extends beyond the feasible region, a row penalty for each row is computed as the dot product of V_i^+ and R^+ plus the dot product of V_i^- and R^- . The row with the greatest row penalty is identified and the revised penalty for that row, if reflected,

is computed. If this reflected penalty is less than the original row penalty, the row is reflected, otherwise it is deleted. When both R^+ and R^- fall within the feasible region, the set of rows which remain constitutes an admissible network set.

III. IMPLEMENTATION OF AUTOMATIC NETWORK IDENTIFICATION HEURISTICS

A. DOUBLE-GUB (D-GUB) ALGORITHM

The D-GUB Algorithm is based on the double-GUB representation of the network problem discussed in the previous section. The automatic GUB algorithm used to find the GUB sets is a gradient method based on a heuristic for approximate solution of certain 0,1 variable linear programming problems developed by Senju and Toyoda [12]. It was coded by Thomen and proved to be the most efficient of the GUB identification algorithms tested in his research. A brief description of the GUB algorithm will be given here; it is fully described in [13].

The GUB heuristic is a two-phase, one-pass, non-back-tracking algorithm which is feasibility seeking. The initial eligible set consists of all rows which can be scaled to the required values according to the scaling scheme employed. Phase 1 attempts to delete as few rows as possible in order to produce a feasible GUB set. Phase 2 examines the rows deleted in Phase 1 and reincludes rows which do not violate the GUB restriction.

The D-GUB algorithm uses a sequential application of the GUB heuristic to the set of network eligible rows. Eligibility for inclusion in the first GUB set is determined by one of the two scaling restrictions described in the previous

section. In the first pass, the initial GUB set is determined. Eligible rows not included in the first GUB set are then examined to determine (under the requirement that two entries in a network column be of opposite sign) if the row is eligible for inclusion in the second GUB set in either present or reflected form. If eligible rows are found, a second pass is made with the GUB algorithm producing a second GUB set.

The D-GUB Algorithm:

Step 0. *Determine Eligible Rows.* Using the scaling scheme desired, determine which rows of the matrix are eligible for selection as network rows.

Step 1. *Find First GUB Set.* Apply the GUB heuristic to the eligible set updating the gradient vector following each row deletion.

Step 2. *Determine Eligibility for Second GUB Set.* For each eligible row not included in the first GUB set, check the columns in which the row has non-zero entries. In each of these columns, if the first GUB set has no non-zero entries or one non-zero entry of opposite sign then the row is eligible for inclusion in the second GUB set in its present form. If the first GUB set has no non-zero entries or a non-zero entry of like sign in each column, then the row is eligible for inclusion in reflected form. Otherwise, the row is not eligible and is discarded.

Step 3. *Find Second GUB Set.* If there are any rows eligible for the second pass, reapply the GUB heuristic to those rows.

Computational experience on a number of real-world models indicates that Phase 2 of the GUB heuristic rarely adds additional rows to the GUB sets obtained in either pass. For the second GUB set, Phase 2 was especially ineffectual. This suggests that the algorithm, which is already extremely fast, can be made even faster by the elimination of Phase 2 with minimal loss of solution quality.

B. NETWORK IDENTIFICATION (NET) ALGORITHM

The NET Algorithm is based on the vector space representation of the network problem discussed in Section II. The algorithm is two-phased, one-pass, and non-backtracking. It is a deletion heuristic which is feasibility seeking. As such, it begins with an eligible set of rows which normally do not form an admissible network set and attempts to delete as few rows as possible to obtain a feasible set. Deleted rows are then considered for reinclusion if they do not violate the feasibility requirement.

The measure of infeasibility at any point is a matrix penalty computed as the sum of individual row penalties. Rows in the eligible set are examined in order of decreasing row penalty and either reflected, if the row penalty would be reduced, or removed and placed in a candidate set for later use. This guarantees that the matrix penalty will be reduced

at each iteration. Thus, the number of iterations in Phase 1 is bounded by the initial matrix penalty, which is polynomially bounded. In Phase 2, the rows in the candidate set are examined for reinclusion in the eligible set if they do not increase the matrix penalty. Those not reincluded are discarded.

Statement of the Problem:

Let $A = \{a_{ij}\}$ be an $m \times n$ matrix with $a_{ij} = 0, \pm 1 \forall i, j$.

Problem: Find a matrix $N = \{n_{ij}\}$ with $(m-k)$ rows and n columns which is derived from A by

1. Deleting k rows of A where $k \geq 0$,
2. Multiplying zero or more rows of A by -1 ,

where N has the property that each column of N has at most one $+1$ element and at most one -1 element.

We wish to find a large N in the sense of containing as many rows as possible, i.e., minimize k .

Terminology and Notation:

1. E is the set of row indices for rows eligible for inclusion in N and is called the eligible set.
2. C is the set of row indices for rows removed from E in Phase I (Deletion). Some rows in C may be readmitted to E in Phase II. C is called the candidate set.
3. The phrase "reflect row i' of A " means to multiply each element in row i' by -1 , i.e., $a_{i',j} \leftarrow -a_{i',j} \forall j$.
4. Other notation will be defined in the algorithm itself.

The NET Algorithm:

Phase I - *Deletion of Infeasible Rows*

Step 0: *Initialization.* Set $E = \{1, 2, \dots, m\}$, $C = \phi$.

For each column j of A compute the + penalty (K_j^+) and the - penalty (K_j^-) as follows:

$$K_j^+ = \left(\sum_{i \in E: a_{ij} > 0} 1 \right) - 1, \quad K_j^- = \left(\sum_{i \in E: a_{ij} < 0} 1 \right) - 1.$$

These penalties represent the number of excess +1 and -1 elements, respectively, in column j which prevent the rows whose indices remain in E from forming a valid N matrix. A penalty value of -1 for K_j^+ (K_j^-) indicates that the column does not contain a +1 (-1) element.

Step 1: *Define Row Penalties.* For every $i \in E$, compute a row penalty (p_i) as follows:

$$p_i = \sum_{j: a_{ij} > 0} K_j^+ + \sum_{j: a_{ij} < 0} K_j^-.$$

This is simply the sum of + penalties for all columns in which row i has a +1 plus the sum of - penalties for all columns in which row i has a -1.

Step 2: *Define Matrix Penalty.* Compute the penalty (h) for the matrix by summing the row penalties as follows:

$$h = \sum_{i \in E} p_i .$$

If $h = 0$, then go to Step 7. Otherwise, go to Step 3.

Step 3: *Row Selection.* Find the row $i' \in E$ with the greatest penalty, i.e.,

$$\text{Find } i' \in E \text{ such that } p_{i'} = \max_{i \in E} p_i .$$

(If there is a tie, choose i' from among the tied values.)

Compute the reflected row penalty $\bar{p}_{i'}$, for i' as follows:

$$\bar{p}_{i'} = \sum_{j: a_{i',j} > 0} (K_j^- + 1) + \sum_{j: a_{i',j} < 0} (K_j^+ + 1) .$$

This would be the row penalty for row i' if it were to be reflected.

Step 4: *Delete, or Reflect Row.*

Case i) $\bar{p}_{i'} \geq p_{i'}$. Let $E \leftarrow E - \{i'\}$, $C \leftarrow C \cup \{i'\}$. Go to Step 5.

Case ii) $\bar{p}_{i'} < p_{i'}$. Reflect row i' . Go to Step 6.

Step 5: *Reduce column penalties* as follows:

For all j such that $a_{i',j} > 0$, $K_j^+ \leftarrow K_j^+ - 1$

For all j such that $a_{i',j} < 0$, $K_j^- \leftarrow K_j^- - 1$

Go to Step 1.

Step 6: *Change column penalties* as follows:

Using the $a_{i',j}$ values after reflection of row i' ,

For all j such that $a_{i',j} > 0$, $K_j^+ \leftarrow K_j^+ + 1$ and $K_j^- \leftarrow K_j^- - 1$

For all j such that $a_{i',j} < 0$, $K_j^+ \leftarrow K_j^+ - 1$ and $K_j^- \leftarrow K_j^- + 1$

Go to Step 1.

Phase II - *Reinclusion of Rows from C*

Step 7. *Eliminate Conflicting Rows.* The rows with indices in E, some possibly reflected from the original A matrix, form a valid N matrix. However, some of the rows removed from E and placed in C may now be reincluded in E if they do not make $h > 0$. Remove from C (and discard) all row indices for rows which, if reincluded in E in present or reflected form, would make $h > 0$. I.e., Remove i from C if

$$\begin{aligned} \text{a) } \exists j_1 \text{ such that } a_{ij_1} > 0 \text{ and } K_{j_1}^+ = 0 \\ \text{or } a_{ij_1} < 0 \text{ and } K_{j_1}^- = 0 \end{aligned}$$

and

$$\begin{aligned} \text{b) } \exists j_2 \text{ such that } a_{ij_2} > 0 \text{ and } K_{j_2}^- = 0 \\ \text{or } a_{ij_2} < 0 \text{ and } K_{j_2}^+ = 0 \end{aligned}$$

If $C = \phi$, STOP, otherwise go to Step 8.

Step 8. *Select Row for Reinclusion.* At this point a row from C may be reincluded in E. There are several possible schemes for selecting the row. After the row is reincluded, the column penalties are adjusted. Then go to Step 7.

No dominating rule has been discovered for breaking ties in maximum row penalty encountered in Step 3. The rule used for the computational results presented herein is to select the row with the minimum number of non-zero entries in an attempt to place a larger number of non-zero entries in the network set. Other possible rules are "first-come, first-served", maximum number of non-zero entries, type of constraint, or modeler preference.

Although the algorithm described above is presented for a matrix with strictly 0,±1 entries, it can be generalized to any matrix by simply letting E be the set of rows with strictly 0,±1 entries or which can be scaled to contain only 0,±1 entries.

Prespecified network rows can also be accommodated with the following modifications:

Let $P = \{i \mid \text{row } i \text{ is prespecified}\}$.

Then $E \leftarrow E - P$.

After computation of K_j^+ and K_j^- in Step 0, for each column j ,

if $\exists i \in P$ such that $a_{ij} = 1$ then $K_j^+ \leftarrow K_j^+ + 1$,

if $\exists i \in P$ such that $a_{ij} = -1$ then $K_j^- \leftarrow K_j^- + 1$.

Rows in P are not eligible for deletion or reflection.

At the termination of the algorithm, the rows in N are given by $E \cup P$.

Computational experience on real-world models indicates that Phase 2 of the NET algorithm is even less productive than that of the GUB algorithm. In only two of sixteen cases were any rows eligible for reinclusion and the maximum number eligible was three. This indicates that the expense of examining the rows in the candidate set for eligibility is probably not justified for the occasional small improvement in quality.

IV. PROBLEM COMPLEXITY

Analysis of the inherent complexity of a problem can reveal whether there is a possibility of developing an efficient algorithm to completely solve all cases of the problem. Unfortunately, analysis of the NET problem indicates that it cannot be solved optimally by an efficient algorithm at this time.

The complexity of a problem is said to be *polynomial* if an algorithm exists which can solve the problem in a number of fundamental operations bounded by a polynomial function of intrinsic problem dimensions. Such an algorithm is called a polynomial time algorithm. An algorithm which is not bounded by such a polynomial function is said to be an *exponential* time algorithm. The maximum solution time for exponential time algorithms grows explosively as problems dimensions increase.

The class of *intractable* problems consists of those problems for which no polynomial time algorithms exist. Between the class of polynomial problems and the class of intractable problems, there is the class of *non-deterministic polynomial* problems--class NP. This class consists of those problems for which a guessed solution can be verified in polynomial time, but for which the existence of a polynomial time algorithm to solve such problems has not yet been proved or disproved.

A problem b is said to be reducible to a problem c if a polynomial time algorithm for solving c (if one exists) can be used to produce, in polynomial time, a polynomial time algorithm for solving b . If every problem of the class NP is reducible to the problem c , then c is said to be *NP-hard*. In addition, if c also belongs to class NP, then c is *NP-complete*. This means that should a polynomial time algorithm be found which solves an NP-complete problem, then polynomial time algorithms exist to solve all problems in class NP. On the other hand, should an NP-complete problem be proved intractable, then all problems in class NP are intractable. The question is still open at this time. For a more complete treatment of complexity theory, see [5], [9].

The problem of finding a GUB set of specified size (i.e., number of rows) was shown by Thomen [13] to be NP-complete, while that of finding a maximum GUB set was shown to be NP-hard. As an observation, the corresponding maximum D-GUB problem with no scaling, since it represents a composition of two disjoint GUB problems, is also NP-complete (for a D-GUB set of specified size) and NP-hard (for a maximum D-GUB set).

The problem of complete conversion of an arbitrary matrix to the node-arc incidence matrix of a pure network, using linear transformation of the matrix as well as row and column scaling, has been shown to be polynomial in complexity [1], [10]. This, however, does not apply to the problem of finding the maximum embedded pure network should complete conversion

fail. Consider the NET problem for a unit matrix, that is, a matrix containing only 0,±1 entries. For the present, no scaling is allowed. A slightly less complex problem than finding the maximum size embedded network set is the following problem:

(NETD) Given an $m \times n$ unit matrix A and an integer $p < m$, determine whether A contains a set of p or more rows such that each column of A (restricted to those rows) has at most two non-zero entries, where entries in the same column must be of opposite sign.

Given a set of p rows from A , it is easy to verify, in polynomial time, whether the set satisfies the above criterion. Given an integer $p < m$, it is not easy to determine whether there exists a set of p or more rows in A which satisfies the criterion. In general, there does not currently exist an algorithm which can do so in polynomial time, as will now be shown.

In a unit matrix, two rows may be said to conflict if they both contain a non-zero element of like sign in a common column. The absence of such pairwise conflicts in a subset of rows from A is not a necessary condition for the rows to form a valid network set if row reflection is allowed. However, that condition is necessary and sufficient for that purpose when no scaling is allowed. With no scaling, it is

evident that the absence of pairwise conflicts is necessary in a valid network set, for the existence of a conflict violates the opposite sign requirement for columns containing two non-zero elements. It is also sufficient, because the violation of the criterion for a valid network set would require at least one column of A to contain at least two non-zero entries of like sign in rows of the set. This, in turn, would imply that the two rows in which this occurs are in conflict.

Following closely the arguments presented by Thomen [13] for the GUB problem, a graph is defined in which the nodes represent the rows of A and two nodes are connected by an edge if and only if the rows conflict in A. The problem of finding a set of p or more rows in A which do not conflict is then equivalent to finding an independent set of size p or more in the graph so defined. This problem, known as the independent set decision problem, is known to be NP-complete [5]. Furthermore, the problem of finding a maximum independent set, and therefore, a maximum GUB set or network set, is NP-hard.

The addition of row reflection to the problem simply means that each row can exist in one of two states, namely, unreflected or reflected. Clearly then, in a set of m rows, there are 2^m distinct states for the set, each corresponding to a different subset of reflected rows. The problem of finding a maximum network set in A, allowing row reflection, is equivalent to

finding a maximum network set with no scaling allowed (shown above to be NP-hard) for 2^m distinct matrices. As a result, this problem is also NP-hard.

Given a general matrix in which non-zero entries may be of any magnitude, and allowing simple row and column scaling, the problem of finding a maximum subset of rows which can be scaled to produce a pure network set can be approached (conceptually) in at least two ways. One approach is to ignore the magnitude of non-zero entries and consider only their sign. When a maximum size network set is found for the resulting unit matrix (an NP-hard problem), an attempt can be made to scale the entries in the rows of the network set to the required $0, \pm 1$ values. The scaling, which may or may not be successful, can be done in polynomial time, however, the entire problem remains NP-hard. Another approach is to consider only subsets of rows which can be scaled to the required unit values should the subset be found to be a valid network set. In essence, the scaling restrictions applied in the algorithms described in this paper guarantee that any subset will have the required unit values. However, if this approach excludes any subsets which may be scalable to the required values, the maximum size set may be missed. Apparently, even with the addition of scaling considerations, the basic problem of finding a maximum embedded pure network set remains NP-hard.

The above analysis of network identification algorithms has only addressed the worst case bound. No conclusions can be made about the average performance of an optimal algorithm. In other words, it may be possible to develop an optimal algorithm with good average performance, but having an exponential worst case bound.

V. UPPER BOUNDS ON MAXIMUM NETWORK SET SIZE

The problem of finding a maximum size pure network set of rows in a matrix, regardless of scaling restrictions, has been shown to be NP-hard. This also applies to the problem of determining the size of a maximum set. Upper bounds on the maximum set size, computed in polynomial time, can be useful in evaluating the quality of network sets produced by heuristic algorithms.

The upper bounds on maximum pure network set size depend on the scaling restrictions employed. It should be noted that the bounds computed here apply to the maximum set size obtainable from the set of eligible rows. They do not apply, in general, to the maximum set size obtainable from the entire coefficient matrix.

Clearly, the maximum set size can be no greater than the number of rows in the eligible set, but this bound is of little practical use. Better bounds can be constructed using information already available in the heuristic procedure. Two upper bounds were developed as a result of this research and are presented here.

The first bound follows directly from the restriction that each column of the matrix (restricted to the eligible set) is allowed at most two non-zero entries. If k represents the maximum number of non-zero entries in any column of A (considering only entries in eligible rows), then it is clear

that at least $k-2$ rows must be deleted from the eligible set in order to make this "worst column" feasible. Since the column counts are readily available in the form of the column penalties (K_j^+ and K_j^-), the upper bound on the network set size for a matrix with m eligible rows is:

$$u_1 = m - \max_j (K_j^+ + K_j^-).$$

This bound is evidently *sharp* in that matrices can be constructed for which it is achieved.

The second bound is tighter, but requires more information about the problem and more computation. It is based on a matrix penalty computed from column penalties, rather than row penalties as in the NET algorithm. This penalty is defined as:

$$H = \sum_{j:K_j^+>0} K_j^+ + \sum_{j:K_j^->0} K_j^- .$$

Clearly, as long as $H > 0$, the rows remaining in the eligible set do not form a valid network set. The reflection of a row in the eligible set may decrease H , increase H , or leave it unchanged. The deletion of a row from the eligible set may decrease H , or leave it unchanged. The actual effect of a reflection or deletion depends on the rows remaining in the eligible set and their state (unreflected or reflected) at the time. However, it is possible to compute for each row

the maximum possible reduction in H obtainable by reflection or deletion of the row, regardless of the other rows remaining in the eligible set. These maximum possible reductions are called the *reflection potential* and *deletion potential* for the row, respectively.

The bound will be computed by determining the minimum number of row deletions necessary to reduce H to zero. This cannot, of course, be computed exactly; however, the result will be conservative in that it will guarantee that at least that number of rows must be deleted.

Consider the possible states of a column j of A in which row i has a non-zero entry (i.e., $a_{ij} \neq 0$). The six possible cases are summarized in Table 1.

Case	K_j^l	K_j^u
1	0	-1
2	0	0
3	0	>0
4	>0	-1
5	>0	0
6	>0	>0

K_j^l = column penalty of like sign to a_{ij}
 (K_j^+ if $a_{ij} > 0$; K_j^- if $a_{ij} < 0$)

K_j^u = column penalty of unlike sign to a_{ij}

Table 1

The non-zero entries in each column are counted only when they occur in the initial eligible set. The penalties used are those computed before any row reflections or deletions have occurred.

Consider first the effect on column j , and thus H , of reflecting row i . In cases 1, 5, and 6, reflection of row i would not change H . In case 4, reflection of row i would decrease H by 1, unless another row with a non-zero in column j was previously reflected. In cases 2 and 3, reflection of row i would actually increase H by 1, unless enough other rows with non-zero entries in column j were reflected or deleted to produce a -1 value for K_j^u . Since we cannot be sure that reflection in cases 2 and 3 would actually increase H , we must consider H unchanged by reflection in these cases. In summary, we allow H to be decreased only by reflection of rows with non-zero entries in columns exhibiting case 4. The reflection potential for row i is computed by summing the effects for each column in which row i has a non-zero element, with the condition that only one row reflection is allowed to decrease H for each column exhibiting case 4.

Row deletions provide greater opportunity for reducing H . In cases 1 and 2, deletion of row i has no effect on H , while in cases 4, 5, and 6, deletion of row i directly decreases H by 1. In case 3, deletion of row i does not directly decrease H , but it allows reflection of another row with a non-zero in column j , producing a net decrease of 1 in the value of

H. In summary, we allow H to be decreased by deletion of rows with non-zero entries in columns exhibiting case 3, 4, 5, or 6. The deletion potential for row i is computed by summing the effects for each column in which row i has a non-zero entry.

To obtain the second bound, the reflection and deletion potentials for each row in the eligible set are computed. Then the maximum possible reduction of H by row reflections alone is computed by summing the individual row reflection potentials. If $H > 0$ at this point, then rows must be deleted. Rows are deleted in order of decreasing deletion potential until $H \leq 0$. The upper bound is then computed as:

$$u_2 = m - \text{number of rows deleted,}$$

where m is the number of rows in the initial eligible set.

This bound is evidently sharp, since examples can be constructed which satisfy the bound exactly.

VI. COMPUTATIONAL RESULTS

The D-GUB and NET algorithms were coded in FORTRAN IV and were tested on 16 large-scale, real-world models. The sizes of the models ranged from small (90 rows, 177 columns) to large (3500 rows, 6500 columns). Table A1 of Appendix A provides the characteristics of each model.

The results obtained for the D-GUB algorithm are given in Table A2 of Appendix A. The row eligibility criterion used for the results reported was that the row contain only 0,±1 entries, or be able to be scaled to 0,±1 entries by row scaling only. The number of eligible rows as a fraction of the total row count ranged from 9% to 100% (the objective row(s) not being eligible in any case). The number of GUB rows obtained in each pass is indicated. In two cases, the entire eligible set was determined to be a GUB set, so no second pass was required. The times given are in CPU seconds for the IBM 360/67 with the program compiled using FORTRAN H (Extended) with code optimization (OPT = 2).

The results for the NET algorithm are given in Table A3 of Appendix A. Also included are the upper bounds on the maximum pure network set size computed from the problem data. The times given for determining the eligible set should be nearly the same as those for the D-GUB algorithm since the same eligibility criterion and code were used in both cases.

The eligibility of rows in the candidate set for reinclusion in Phase 2 was determined, but Phase 2 was not included due to the absence of eligible rows in nearly every case. The solution time does not include the time required to determine eligibility for Phase 2. The times given are again in CPU seconds on the IBM 360/67. The NET quality value is the number of rows in the network set, expressed as a percentage of the best known upper bound on the pure network set size. As explained earlier, the actual maximum network set size is, in general, unknown and thus the actual NET quality may be better than this conservative estimate. In particular, the bounds are almost certainly too high for problems with a large number of eligible rows (e.g., PAPER) and for problems with dense, unstructured coefficient matrices (e.g., TRUCK).

The overall results obtained are very encouraging. The algorithms are very fast (especially when compared with computer time expended in any attempt to solve these large problems) and they consistently produce maximum or near maximum pure network sets (from the eligible rows) as evidenced by the upper bounds. Work is underway to include one or both algorithms as a part of a state-of-the-art optimization system which will provide results concerning the value of the factorizations obtained in improving the solution times for the models.

VII. EXTENSIONS

The current weakness in this research appears to lie primarily in the area of scaling. Many problems exhibiting intrinsic network structure are disguised by their formulation and resist the simplistic attempts used here to rescale them. In particular, the COAL model is known to be a complete network, if appropriately scaled, but it is not evident how this is to be discovered using general, problem-independent automatic identification. Methods used to scale an entire matrix to 0,±1 values (see [1], [10]) can be attempted, but failing complete conversion the next step is not evident. This remains a potentially fruitful area for further research.

Another potentially profitable area is the extension of the methods developed for the network identification and associated upper bounds to other types of special structure. Schrage [11] has reported research in this area using the conflict method of Greenberg and Rarick [7]. It is felt that the superior speed of the gradient method exhibited in the GUB factorization [13] and seen here in the network factorization would carry over to the identification of other special structures. This approach is currently being pursued.

Finally, the possible use of the automatic identification algorithms in discovering special structure which can be used in the automatic decomposition of very large models looks extremely promising. If the algorithms can be used

to identify subproblems connected by a few coupling constraints, then a decomposition of the problem into smaller problems, each with a special structure (e.g., transportation or assignment problems), may be possible. The solution of these smaller problems using very efficient, specialized algorithms may assist in the solution of the larger problem. One class of such models is that of large multi-commodity production-transportation problems.

VIII. CONCLUSIONS

The stated purpose of this research was to extend the results obtained for automatic GUB identification in large-scale models to embedded pure network identification. As was the case with the GUB problem, the maximum embedded pure network identification was shown to be NP-hard in complexity. Therefore, heuristic algorithms were developed and used to identify large pure network sets in polynomial time.

With regard to accomplishing the intended purpose, the research was very successful. Using restrictions on row eligibility to overcome the inherent scaling problems, the algorithms proved extremely efficient in identifying large pure network sets. Upper bounds were developed for maximum pure network set size and were used to evaluate the effectiveness of the algorithms. The results on problem complexity were obtained and, as expected, justify the use of heuristic algorithms to approximate optimal solution of what would otherwise be an extremely difficult problem.

The unexpected benefit of this research appears to be the applicability of both the general NET identification procedure and the upper bound computations to many other types of special structures. Research is continuing in this area.

APPENDIX A

This appendix contains three tables describing the linear and mixed integer programming models used to test the algorithms described in this paper and the computational results of the algorithms applied to the models. The content of the tables is:

TABLE A1: Sample LP (MIP) Model Characteristics

TABLE A2: D-GUB Algorithm Computational Results

TABLE A3: NET Algorithm Computational Results

The execution times provided in the tables are expressed in CPU seconds on an IBM 360/67 with the programs compiled using FORTRAN IV H (Extended). Further description of the data in the tables is provided in Section VI, Computational Results.

TABLE A1

SAMPLE LP (MIP) MODEL CHARACTERISTICS

<u>MODEL</u>	<u>DESCRIPTION</u>	<u>ROWS</u>	<u>COLUMNS</u>		<u>NON-ZERO COEFFICIENTS</u>
			<u>TOTAL</u>	<u>INTEGER</u>	
TRUCK	Fleet Dispatch Model	220	4,752	4,752	30,074
FOAM	Production Scheduling Model	1,000	4,020	42	13,083
AIRLP	Fleet Dispatch Model	171	3,040	0	6,023
ELEC	Energy Prod. & Consump. Model	785	2,800	0	8,462
ODSAS	Manpower Planning Model	4,648	4,683	0	30,520
LANG	Equip. & Manpower Sched. Model	1,236	1,425	0	22,028
FERT	Production & Distribution Model	606	9,024	0	40,484
COAL	Energy Development Model	171	3,753	0	7,506
CUPS	Production Scheduling Model	361	582	145	1,341
PAD	Energy Prod. & Consump. Model	695	3,934	0	13,459
JCAP	Production Scheduling Model	2,487	3,849	560	9,510
PAPER	Production Allocation Model	3,529	6,543	0	32,644
NETTING	Currency Exchange Model	90	177	114	375
PIES	Energy Prod. & Consump. Model	663	2,923	0	13,288
GAS	Production Scheduling Model	799	5,536	0	27,474
PILOT	Energy Prod. & Consump. Model	976	2,172	0	13,057

TABLE A2

D-GUB ALGORITHM RESULTS

MODEL	ELIGIBILITY		NETWORK ROWS FOUND				TOTAL TIME	ROWS REFLECTED	NONZEROS IN SET	
	ROWS	TIME	PASS 1 TIME	PASS 2 TIME	ALL GUB	OTHER				
TRUCK	219	0.76	29	3.96	18	4.44	47	8.40	18	1,755
FOAM	966	0.33	917	0.95	34	0.94	951	1.89	1	8,001
AIRLP	150	0.09	150	0.41	ALL	GUB	150	0.41	0	3,000
ELEC	322	0.12	266	0.47	6	0.52	272	0.99	6	2,691
ODSAS	410	0.61	195	1.84	122	1.55	317	3.39	92	5,344
LANG	850	0.26	342	2.91	243	0.83	585	3.74	1	1,804
FERT	585	0.62	559	3.00	13	3.03	572	6.03	13	16,291
COAL	111	0.11	111	0.50	ALL	GUB	111	0.50	0	3,753
CUPS	300	0.05	150	0.14	101	0.15	251	0.29	1	710
PAD	174	0.10	160	0.52	0	0.06	160	0.58	0	1,552
JCAP	1,811	0.26	517	1.49	357	1.01	874	2.50	201	2,622
PAPER	2,324	0.79	1,016	3.53	468	3.71	1,484	7.24	433	8,176
NETTING	59	0.01	36	0.03	18	0.04	54	0.07	18	89
PIES	142	0.09	128	0.51	0	0.05	128	0.56	0	1,392
GAS	752	0.58	607	2.61	75	2.39	682	5.00	75	9,008
PILOT	109	0.10	96	0.44	13	0.48	109	0.92	1	479

TABLE A3

NET ALGORITHM RESULTS

MODEL	ELIGIBILITY		UPPER BOUNDS		NETWORK ROWS FOUND			ROWS REFLECTED	NONZEROS IN SET
	ROWS	TIME	U1	U2	NUMBER	TIME	QUALITY		
TRUCK	219	0.76	214	137	46	19.82	33.58%	18	1,781
FOAM	966	0.34	965	955	951	1.16	99.58%	1	8,001
AIRLP	150	0.09	150	150	150	0.35	100%	0	3,000
ELEC	322	0.14	310	306	286	2.07	93.46%	34	2,915
ODSAS	410	0.61	406	369	286	14.55	77.51%	45	6,207
LANG	850	0.29	836	758	661	14.82	87.20%	2	2,239
FERT	585	0.62	584	572	572	6.15	100%	13	16,291
COAL	111	0.11	111	111	111	0.43	100%	0	3,753
CUPS	300	0.05	299	297	295	0.14	99.33%	1	862
PAD	174	0.10	171	164	160	0.59	97.56%	0	1,552
JCAP	1,811	0.26	1,801	1,092	917	44.07	83.97%	200	2,540
PAPER	2,324	0.66	2,316	2,072	1,627	94.16	78.52%	603	8,995
NETTING	59	0.01	58	57	54	0.08	94.74%	18	89
PIES	140	0.09	139	132	128	0.59	96.97%	0	1,392
GAS	752	0.60	750	710	668	9.71	94.08%	33	11,002
PILOT	109	0.10	109	109	109	0.36	100%	1	479

LIST OF REFERENCES

1. Bixby, R. E. and Cunningham, W. H., "Converting Linear Programs to Network Problems," (to appear in Mathematics of Operations Research).
2. Brearly, A. L., Mitra, G., and Williams, H. P., "Analysis of Mathematical Programming Models Prior to Applying the Simplex Algorithm," Mathematical Programming, v. 8, p. 54-83, 1975.
3. Brown, G. G. and Thomen, D. S., "Automatic Factorization of Generalized Upper Bounds in Large Scale Optimization Models," Technical Report NPS55-80-003, Naval Postgraduate School, Monterey, January 1980.
4. Dantzig, G. B. and Van Slyke, R. M., "Generalized Upper Bounding Techniques," Journal of Computer and System Sciences, v. 1, p. 213-226, 1967.
5. Garey, M. R. and Johnson, D. S., Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman and Company, 1979.
6. Graves, G. W. and McBride, R. D., "The Factorization Approach to Large-Scale Linear Programming," Mathematical Programming, v. 10, p. 91-110, 1976.
7. Greenberg, H. J. and Rarick, D. C., "Determining GUB Sets via an Invert Agenda Algorithm," Mathematical Programming, v. 7, p. 240-244, 1974.
8. McBride, R. D., "Linear Programming with Linked Lists and Automatic Guberization," Working Paper No. 8175, University of Southern California, School of Business, July 1975.
9. Klee, V., "Combinatorial Optimization: What is the State of the Art," Mathematics of Operations Research, v. 5, p. 1-26, February 1980.
10. Musalem, J. S., "Converting Linear Models to Network Models," Ph.D. Dissertation, UCLA, January 1980.
11. Schrage, L. (Private Communication, March 1980).

12. Senju, S. and Toyoda, Y., "An Approach to Linear Programming with 0-1 Variables," Management Science, v. 15, p. B196-B207, 1968.
13. Thomen, D. S., "Automatic Factorization of Generalized Upper Bounds in Large Scale Optimization Problems," M.S. Thesis, Naval Postgraduate School, September 1979.
14. Wright, W. G. and Brown, G. G., "Automatic Factorization of Embedded Structure in Large-Scale Optimization Models," Proceedings of the Symposium on Computer-Assisted Analysis and Model Simplification, Boulder, Colorado, March 1980.

INITIAL DISTRIBUTION LIST

	No. Copies
1. Defense Technical Information Center Cameron Station Alexandria, Virginia 22314	2
2. Library, Code 0142 Naval Postgraduate School Monterey, California 93940	2
3. Department Chairman, Code 55 Department of Operations Research Naval Postgraduate School Monterey, California 93940	1
4. Professor Gerald Brown, Code 55BW Naval Postgraduate School Monterey, California 93940	1
5. Professor Glenn W. Graves Graduate School of Management University of California Los Angeles, California 90024	1
6. CAPT William G. Wright, USMC 1508 Mountain View Tacoma, Washington 98465	1

Thesis
W916
c.1

Wright

189527

Automatic identifica-
tion of network rows in
large-scale optimization
models.

Thesis
W916
c.1

Wright

189527

Automatic identifica-
tion of network rows in
large-scale optimization
models.

thesW916

Automatic identification of network rows



3 2768 000 98879 4

DUDLEY KNOX LIBRARY