AUTOMATIC IDENTIFICATION OF EMBEDDED STRUCTURE IN LARGE-SCALE OPTIMIZATION MODELS

Gerald G. Brown
William G. Wright

Naval Postgraduate School
Monterey, California

This paper discusses automatic detection and exploitation of embedded structure in Large-Scale Linear Programming (LP) models. We report experiments with real-life LP and mixed-integer (MIP) models in which various methods are developed and tested as integral modules of an optimization system of advanced design [6]. We seek to understand the modeling implications of these embedded structures as well as to exploit them during actual optimization. The latter goal places heavy emphasis on efficient, as well as effective, identification techniques for economic application to large models. Several (polynomially complex) heuristic algorithms are presented from our work. In addition, bounds are developed for the maximum row dimension of the various factorizations. These bounds are useful for objectively estimating the quality of heuristically derived structures.

I. INTRODUCTION

Automatic detection and exploitation of special structure in large-scale LP (or MIP) models has been the subject of a continuing research program conducted at the Naval Postgraduate School and UCLA over the past decade. This paper draws from various results in this effort, and refers (sparingly) to significant work by other researchers. The references contain complete descriptions of these results for the interested reader.

ISBN 0-12-299680-1

Our scope is intentionally limited to automated methods of sufficient efficiency to enable us to economically apply them to real-world optimization problems. Thus, we consider only those approaches showing greatest promise for immediate practical application. Although the interpretations of embedded model structure can lend profound insights in their own right, we are equally interested in detecting errors in data preparation and model generation -- mathematically mundane issues of fundamental importance to the practitioner.

The sheer size of contemporary large-scale LP models presents significant computational difficulties, even for otherwise elementary factorizations. Implementation of effective structural analysis procedures is primarily a matter of exercising large-scale data structures efficiently. As we shall see, though, these practical considerations can give significant theoretical guidance in the specification of efficiently achievable classes of model transformations.

That detection of embedded special structure can be of practical importance in actual model solution is undisputed. It is widely known that explicit simplex operations can be materially improved in efficiency by incorporation of basis factorization methods (e.g., [6], [9], and references of [7]). The details of such modifications of the simplex procedure are not given here. However, the underlying themes of simplex factorization are the substitution of logic for floating point arithmetic, and separation of the apparent problem monolith into more manageable components.

This paper deals exclusively with row factorizations. The pervasive implied problem for row factorization is the identification of the *best* embedded structure from all those that may lie at hand in any particular model. Conventional wisdom differs as to the criterion for this discrimination among factorizations of the same class. However, it is generally accepted that the row dimensionality of the factorization serves as an excellent measure of effectiveness. In this sense, embedded special structures fall naturally into a taxonomy implied by the intrinsic complexity of the associated maximum row identification problems.

We proceed with a discussion of several types of embedded special structures detectable by efficient polynomially complex algorithms. These structures are considered in increasing order of maximum row identification complexity. We emphasize that *efficient* polynomial algorithms are operationally defined here as low-order polynomial in terms of intrinsic problem dimensions (e.g., number of rows, columns, and non-zero elements), and *not* in terms of the total volume of model information (e.g., total number of bits in all coefficients, ad nauseam).

2. SIMPLE REDUCTIONS

LP models often exhibit simply detected structural characteristics which permit a reduction in row dimensionality without loss of model information. Several such reductions are possible in evidently polynomial complexity. These include:

a) Void Rows
b) Void Columns
c) Singleton Rows (simple upper bounds)
d) Singleton Columns
e) Fixed Variables
f) Rows that Fix Variables
g) Null Variables
h) Non-extremal Variables
i) Redundant Rows.

Some of these reductions do not obviously decrease row dimension. However, the reductions may be applied repeatedly to the model, revealing at each iteration more rows which can be removed. Thus the cyclic application of reductions continues until a minimal model results.

Experiments with some of these reductions have been reported by Brearley, Mitra and Williams [5]. More extensive work at large-scale has been done by Bradley, Brown and Graves [3] and by Krabek [11].

Detection of *all* redundant LP rows requires complete solution of equivalent LP problems, and is thus equivalent in complexity to LP. (We hesitate to say polynomial in the sense of Khachian's recent result.) Thus, we restrict redundant row detection to *orthogonal* redundancy, revealed by substitution of bounds for problem variables. An efficient detection algorithm results.

With real-life LP and MIP models, a remarkably large fraction of model rows can be removed by these simple techniques. For some cases, models have been nearly *solved* this way.

We note that integrality conditions can be superimposed on these simple reductions (e.g., tighten bounds on integer variables by truncation) to strengthen them. Nonlinear models also benefit from these reductions, and from others not addressed in this paper.

Table 1 contains the characteristics of several real-life linear and mixed integer models. Table 2 displays the results of simple reductions applied to these models [3]. Multiple *passes* are made for each model until no more reductions are possible. The times given are for execution on an IBM 360/67 using FORTRAN H (Extended) without code optimization.

3. GENERALIZED UPPER BOUNDS

Rows for which each column has at most one non-zero coefficient (restricted to those rows) collectively form a generalized upper bound (GUB) set. Usually, we additionally require that

TABLE 1. Sample LP (MIP) Models

Model	Rows	Columns: Total	Columns: Integer	Non-zero Coefficients
TRUCK	220	4,752	4,752	30,074
FOAM	1,000	4,020	42	13,083
AIRLP	171	3,040	0	6,023
ELEC	785	2,800	0	8,462
ODSAS	4,648	4,683	0	30,520
LANG	1,236	1,425	0	22,028
FERT	606	9,024	0	40,484
COAL	171	3,753	0	7,506
CUPS	361	582	145	1,341
PAD	695	3,934	0	13,459
JCAP	2,487	3,849	560	9,510
PAPER	3,529	6,543	0	32,644
NETTING	90	177	114	375
PIES	663	2,923	0	13,288
GAS	799	5,536	0	27,474
PILOT	976	2,172	0	13,057

the coefficients in these rows be capable of being rendered to ± 1 by simple row or column scaling.

The problem of identifying a GUB set of *maximum* row dimension is NP-hard [7], making optimal GUB factorization algorithms hopelessly inefficient for our purposes. Heuristics adapted from work by Graves and by Senju and Toyoda (see [14], and references of [5] and [7]) work very effectively and dependably at large-scale to find *maximal* GUB sets.

Unfortunately, the problem of determining just the *size* of the maximum GUB set is also NP-hard. However, Brown and Thomen [7] have developed bounds on the size of the maximum GUB set which are sharp and easily computed. These bounds have been used to show, in some cases, that maximum GUB sets have been

TABLE [illegible]. Simple Reductions [3]

Model	Fixed Columns	Singleton Columns	Singleton Rows	Doubleton Equations	Redundant Rows	Passes	CPU Sec.
[illegible]	2	0	0	0	1	2	5.57
[illegible]	2	0	36	0	0	2	3.30
[illegible]	20	0	0	0	0	2	1.78
[illegible]	494	56	120	3	14	4	8.64
[illegible]	0	40	0	3,609	40	3	31.00
[illegible]	105	220	68	9	55	20	61.45
[illegible]	406	0	0	0	13	4	14.25
[illegible]	0	0	0	0	0	2	2.12
[illegible]	67	49	18	39	55	4	1.90
[illegible]	133	30	16	0	0	3	3.26
[illegible]	6	414	277	180	360	3	12.16
PAPER	145	190	90	359	45	5	20.61
NETTING	8	1	29	7	17	4	0.81
[illegible]	133	50	16	0	0	3	3.32
GAS	501	60	31	0	30	4	10.08
PILOT	277	123	12	36	91	11	17.15

TABLE 3. GUB Factorization [7]

Model	Rows-GUB Eligible	Row Conflicts		GUB		
		Count	Density	Rows	Quality	SEC
TRUCK	219	10,436	43.53%	29	20.28%	5.00
FOAM	989	8,186	1.67%	917	98.18%	1.73
AIRLP	170	2,983	20.64%	150	100.00%	0.65
ELEC	784	6,167	2.01%	309	62.80%	1.15
CDSAS	4,647	5,220	0.05%	749	18.61%	7.12
LANG	1,235	46,424	6.09%	342	35.15%	14.90
FERT	605	16,455	9.01%	559	93.59%	6.73
COAL	170	3,753	26.13%	111	91.74%	0.92
CUPS	336	744	1.32%	160	66.67%	0.21
PAD	694	4,416	1.84%	188	41.87%	3.34
JCAP	2,446	16,578	0.55%	529	29.19%	2.23
PAPER	3,528	35,047	2.82%	1,041	34.65%	5.77
NETTING	71	46	1.85%	36	78.26%	0.05
PIES	662	4,116	1.88%	172	40.76%	2.82
GAS	789	22,220	7.15%	608	93.25%	3.79
PILOT	975	12,110	2.55%	255	33.73%	2.75

achieved via heuristic methods. In any case, the bounds provide excellent objective measure of the quality of any GUB set, regardless of the means of its derivation. Frequently, manual GUB analysis will suffice for models with amenable structure.

The bounds are developed in terms of the number of distinct *conflicts* present in the model. Two rows are in conflict if they each have a non-zero element in a common column, making them mutually exclusive in a GUB set. If s_i is the number of rows in conflict with row i, then the total problem conflict count for a model with m rows is

$$c = \frac{1}{2}\sum_i s_i < \frac{1}{2} m(m-1) \ .$$

A problem-independent bound on the size of the maximum GUB set is [7]

$$u_1 = \lfloor .5 + \sqrt{.25 + m(m-1) - 2c} \ ,$$

where $\lfloor$ indicates truncation to an integer.

A tighter, problem-dependent bound is

$$u_2 = \begin{cases} m - \lceil \frac{c}{y} \ , & c \leq (m-y)y \\ \lfloor .5 + \sqrt{.25 + y(2m-y-1) - 2c} \ , & c > (m-y)y \ ; \end{cases}$$

where

$$y = \max_i s_i \ .$$

Tighter upper bounds have been derived for the size of the maximum GUB set, as well as lower bounds.

Table 3 contains the results of automatic GUB factorization applied to the benchmark models [7]. Row eligibility is based on the capability to scale the row to contain only 0, ± 1 coefficients. *GUB quality* is the number of GUB rows found, expressed as a percentage of the best known upper bound on maximum GUB row dimension (actual GUB quality may be better than this conservative estimate). The results were obtained using FORTRAN H (Extended) with code optimization.

4. IMPLICIT NETWORK ROWS

Implicit network rows are a set of rows for which each column has at most two non-zero coefficients (restricted to those rows) and for which columns with two non-zero coefficients (in those rows) can be converted by *simple* row and column scaling such that the non-zero coefficients have opposite sign. Such rows in LP are commonly called networks with gains.

Pure network rows (NET) can be converted by *simple* row and column scaling such that all non-zero coefficients (restricted to those rows) have value ± 1, and such that columns with two non-zero coefficients (in those rows) have one $+1$ and one -1. Such rows in LP are called pure networks (e.g., [4]).

Simple row and column scaling is restricted such that application of each scale factor renders an entire row, or column, to the desired sign (and unit magnitude for pure NET).

The problem of identifying a NET factorization of *maximum* row dimension is NP-hard [15], making optimal NET identification algorithms practically useless. The problem of determining just the *size* of the maximum NET set is also NP-hard. Thus, heuristic identification methods are mandated.

An extension of GUB heuristics can be used to achieve NET factorizations. First, a GUB set is determined by methods mentioned in Section 3. Then, a second GUB set is found from an eligible subset of remaining rows. The second GUB set is conditioned such that its row members must possess non-zero coefficients of opposite sign in each column for which the prior GUB set has a non-zero coefficient.

This double-GUB (DGUB) factorization yields a *bipartite* NET factorization. Thus, DGUB heuristically seeks the maximum embedded transportation or assignment row factorization. Pure network equivalents derive from proper editing of eligible rows.

Generalizing on the theme of Senju and Toyoda [14], a more general method has been developed by Brown and Wright [8] for direct NET factorization of implicit network rows. Pure NET rows can be identified with the same procedure by simple screening of admissible candidate rows.

This heuristic is designed to perform a network factorization of a signed elementary matrix $(0,\pm 1$ entries only). It is a deletion heuristic which is feasibility seeking. The measure of infeasibility at any point is a matrix penalty computed as the sum of individual row penalties. The algorithm is two-phased, one pass, and non-backtracking. The first phase yields a feasible set of rows, while the second phase attempts to improve the set by reincluding rows previously excluded. Each iteration in Phase I either deletes a row or reflects it (multiplies it by -1) and guarantees that the matrix penalty will be reduced. Thus, the number of iterations in Phase I is bounded by the initial value of the matrix penalty, which is polynomially bounded.

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 phase "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.

Algorithm

Phase I - Deletion of Infeasible Rows

Step 0: Initialization. Set $E = \{1,2,\ldots,m\}$, $C = \phi$. 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 - Reintroduction 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:

a) $\exists\ j_1$ such that $a_{ij_1} > 0$ and $K^+_{j_1} = 0$

or $a_{ij_1} < 0$ and $K^-_{j_1} = 0$.

and

b) $\exists\ j_2$ such that $a_{ij_2} > 0$ and $K^-_{j_2} = 0$

or $a_{ij_2} < 0$ and $K^+_{j_2} = 0$.

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.

Modifications can be made to Step 0 to allow for 1) Matrices including non- $0, \pm 1$ entries and/or 2) Prespecified network rows. The modifications are:

1. $E = \{i \mid a_{ij} = 0, \pm 1 \text{ for all } j\}$.
2. Let $P = \{i \mid \text{row } i \text{ is prespecified}\}$.

   $E \leftarrow E - P$

   After computation of K^+_j and K^-_j, find for all 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 $K^-_j \leftarrow K^-_j + 1$.

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

One easily obtained upper bound on the maximum row dimension of the network factorization is:

$$\ell_j = m - \max_j (K^+_j + K^-_j) .$$

This bound is easily computed and evidently sharp. It can be used to objectively evaluate the quality of a heuristically derived network factorization. The bound may also be used to preemptively terminate factorization effort.

Other bounds may be similarly derived.

Table 4 displays the results of DGUB and NET factorizations of the benchmark models. Row eligibility is determined by the capability to scale each row, by row scaling alone, to contain only $0, \pm 1$ entries. The *NET quality* is the number of NET rows found, expressed as a percentage of the upper bound on maximum NET row dimension given above (actual NET quality may be considerably better than this estimate).

5. HIDDEN NETWORK ROWS

Hidden network rows[1] are a set of rows which satisfy NET row restrictions after linear transformation of the model. That is, realization of these (LNET) rows may require a general linear transformation of the original model.

The discrimination between *implicit* and *hidden* network rows is not (necessarily) in their use, but rather in their detection. The transformation group associated with implicit network rows involves *only* permutations and simple scaling of individual rows and columns. The hidden network rows require a completely general linear transformation and partial ordering. Thus, identification of hidden networks requires significant computation just to identify eligible rows, since any given row may conflict with subsets of its cohorts after transformation.

This problem has been solved for *complete* hidden network factorization, where all rows are shown to be LNET or the algorithm fails. Bixby and Cunningham [2] and Muslem [13] have

[1]*We have adopted the term* hidden *from Bixby [1], but the definition may not universally appear to be equivalent.*

TABLE 4. NET Factorization [8]

Model	Rows NET Eligible	DGUB Rows	DGUB SEC	NET Rows	NET Quality	NET SEC
TRUCK	219	47	8.40	46	33.58%	19.83
FOAM	966	951	1.89	951	99.58%	1.16
AIRLP	150	150	0.41	150	100.00%	0.35
ELEC	322	272	0.99	286	93.46%	2.07
ODSAS	410	317	3.39	286	77.51%	14.55
LANG	850	585	3.74	661	87.20%	14.82
FERT	585	572	6.03	572	100.00%	6.15
COAL	111	111	0.50	111	100.00%	0.43
CUPS	300	251	0.29	295	99.33%	0.14
PAD	174	160	0.58	160	97.56%	0.59
JCAP	1,911	874	2.50	917	83.97%	44.07
PAPER	2,324	1,484	7.24	1,627	78.52%	94.16
NETTING	59	54	0.07	54	94.74%	0.08
PIES	142	128	0.56	128	96.97%	0.59
GAS	752	682	5.00	668	94.08%	9.71
PILOT	109	109	0.92	109	100.00%	0.36

given polynomially complex methods for complete LNET conversion. (The complete GUB problem is polynomial as well.)

Strategically, the complete hidden LNET factorization requires two steps:

Detection: necessary conditions for existence of a complete LNET factorization must be established, and

Scaling: a linear transformation to achieve the NET structure must be determined, if one exists.

Cunningham and Bixby attempt detection, followed by scaling. Musalem tries scaling, then detection. This is a crucial difference between methods, since problems which cannot be completely NET factorized may fail in either step.

Briefly, Cunningham and Bixby *detect* by showing that the incidence matrix of the model rows can be converted to a graphic matroid. They employ a method by Tutte (see references of [2]). Given success, the graphic record of the detection is used to attempt to *scale* the model to NET, or to show that no such scaling exists.

Musalem *scales* the model to a ± 1 matrix, and then uses a method by Iri (see references of [13]) to build a tree, edge by edge, which reveals the partial ordering coincident with complete hidden LNET factorization.

Both methods are polynomially complex. However, complete LNET factorization is relatively expensive by either method in that quite a large amoung of real arithmetic and logic is required. Underlying data structures have not been suggested for either method. Both methods fail if complete LNET factorization is impossible, and neither leaves the investigator with much information useful in salvaging a partial LNET factorization. We conjecture that risk of preemptive failure narrowly favors the Musalem approach, since he defers the relatively involved detection step.

Locating a hidden LNET factorization of *maximal* row dimension has been suggested by Bixby [1] and by Musalem [13], but no concrete method is given and no computational testing is reported. Evidently, the *maximum* LNET problem is NP-hard, and its maximal relaxation remains unsolved in the practical sense of this report.

6. CONCLUSION

The techniques reported here have been used with great success on a wide variety of large LP (MIP) models. The context of this research is certainly atypical in that the models which we work with are often sent to us for analysis and solution precisely because they have already failed elsewhere. In these cases, our motives are to quickly diagnose suspected trouble before optimization, prescribe remedies, and perform the actual optimization reliably and efficiently.

This has undoubtedly biased our view of structural detection methods. Practical considerations arising from turnaround deadlines and the specific advantages of our own optimization system [6][2] have colored our judgment. Many provocative suggestions for further research have not been pursued, either due to lack of opportunity, to poor intuition, or to sheer economics. Whether or not by equivalent prejudice, Krabek [11] reports some similar methods for simple reductions applied to large MIP's.

A great deal of insight has been gained from these experiments. The cost of factorization is truly insignificant

[2]*The* X-System *(XS) differs in many ways from classical large-scale mathematical programming systems; it simultaneously supports simple and generalized upper bounds, general basis factorization, MIP, nonlinear, and decomposition features. In addition, the fundamental LP algorithm has been enhanced to intrinsically incorporate* elastic *range restrictions. XS is particularly suited for solution in limited time of large models with complicating features.*

relative to the information and (primarily) solution efficiency gained thereby. Revelations have ranged from outright rejection of absurd formulations to subtle inferences on the interpersonal conflicts of model proponents. Very few models fail to reveal some totally unsuspected structural curiosity. Indeed, it is often some minor aberration that proves most revealing. Sometimes, the combined effects of several minor features collectively contribute to a discovery of significant model attributes.

Our general operational guidelines has been to avoid heavy computational investment in factorization. Rather, highly efficient methods are used *repeatedly* on variations of each model. Manual and *intuitive* analysis of these results usually reveal much more than could be reasonably expected from any totally automated method applied to problems of exponential complexity. Interactive analysis of large-scale models is uncompromisingly challenging in a technical sense and equally rewarding.

Accordingly, we have not yet implemented maximal hidden network heuristics, or block-angular clustering methods. In the former case, we find intrinsic NET factorization to unerringly reveal more general network forms. Also, reformulation to a NET factorization commonly requires more than a linear transformation; variables and constraints must frequently be *augmented* to achieve the desired arc and node interpretation.

In the case of block-angular and attendant structures, we require a great deal more information than row and column index subsets and aggregate relations to develop an effective and economically sensible mathematical decomposition scheme; further; even for unfamiliar models such structure is usually apparent in those cases that invite decomposition.

For a more circumspect and less mechanical review of structural interpretation for LP models, see Greenberg [10]. From the standpoint of his paper, the techniques reported here are simply monadic functions of analysis.

Large factorizations are routinely found as intrinsic features in real-life models. However, we feel that it is an abominable practice to proselytize in favor of some particular model structure at the expense of model realism or common sense.

For instance, network models have recently received unprecedented attention in the literature. The implication has often been that since networks are usually found in models, networks should be used as the exclusive model. This is, of course, patent nonsense, smacking of a solution in search of a problem. An analyst should view factorizations as specializations of models, rather than forcing models to fit certain popular factorizations [4].

7. ACKNOWLEDGMENTS

We are deeply indebted to our colleagues Gordon Bradley and Glenn Graves, who have contributed fundamentally to this research. David Thomen is largely responsible for the GUB identification material.

8. REFERENCES

[1] Bixby, R., "Hidden and Embedded Structure in Linear Programs," paper presented at the Symposium in Computer-Assisted Analysis and Model Simplification, Boulder, 24 March 1980.

[2] Bixby, R. and W. Cunningham, "Converting Linear Programs to Network Problems," c. October, 1979 (to appear in *Mathematics of Operations Research*).

[3] Bradley, G., G. Brown, and G. Graves, "Preprocessing Large Scale Optimization Models," paper presented at ORSA/TIMS, Atlanta (Nov. 1977).

[4] Bradley, G., G. Brown, and G. Graves, "Design and Implementation of La[illegible] Transshipment Algorithms," [illegible], No. 1, (1977), p. 1.

[5] Brearly, A., G. Mitra, and H. Williams, "Analysis of Mathematical Programming Problems Prior to Applying the Simplex Algorithm," *Mathematical Programming 8*, (1978), p. 54.

[6] Brown, G., and G. Graves, "Design and Implementation of a Large-Scale (Mixed Integer) Optimization System," paper presented at ORSA/TIMS, Las Vegas, Nov. 1975.

[7] Brown, G., and D. Thomen, "Automatic Factorization of Generalized Upper Bounds in Large Scale Optimization Models," Technical Report NPS55-80-003, Naval Postgraduate School, Monterey (January 1980).

[8] Brown, G., and W. Wright, "Automatic Identification of Network Rows in Large-Scale Optimization Models," (in preparation).

[9] Graves, G., and R. McBride, "The Factorization Approach to Large-Scale Linear Programming," *Mathematical Programming 10*, (1976), p. 91.

[10] Greenberg, H., "Implementation Aspects of Model Management: A Focus on Computer-Assisted Analysis," (Nov. 1979).

[11] Krabek, C., "Some Experiments in Mixed Integer Matrix Reduction," paper presented at XXIV International TIMS Meeting, Honolulu, June 18, 1979.

[12] McBride, R., "Factorization in Large-Scale Linear Programming," Ph.D. Dissertation, UCLA (1973).

[13] Musalem, S., "Converting Linear Models to Network Models," Ph.D. Dissertation, UCLA (Dec. 1979).

[14] Senju, S., and R. Toyoda, "An Approach to Linear Programming with 0-1 Variables," *Management Science 15*, (1968), p. B196.

[15] Wright, W., "Automatic Identification of Network Rows in Large-Scale Optimization Models," M.S. Thesis, Naval Postgraduate School (May 1980).