Working Paper No. 120

IMPLICIT ENUMERATION

USING AN IMBEDDED LINEAR PROGRAM

by

ARTHUR M. GEOFFRION

May, 1967

WESTERN MANAGEMENT SCIENCE INSTITUTE
University of California, Los Angeles

University of California

Los Angeles

Western Management Science Institute

Working Paper No. 120

IMPLICIT ENUMERATION USING AN IMBEDDED LINEAR PROGRAM

by

072
327

Arthur M. Geoffrion

May 1967

This working paper should be regarded as preliminary in nature, and subject to change before publication in the open literature. It should not be quoted without prior consent of the author. Comments are cordially invited.

This work was jointly sponsored by the United States Air Force under Project RAND, and by the Western Management Science Institute under grants from the National Science Foundation and the Office of Naval Research. It is a pleasure to acknowledge the computational assistance provided by R. J. Clasen and A. B. Nelson. The present paper develops and implements results obtained in an earlier unpublished paper by the author [6].

SUMMARY

Integer programming by implicit enumeration has been the subject of several recent investigations. Computational efficiency seems to depend primarily on the ability of various tests, applied to the constraints in connection with "partial solutions," to exclude from further consideration a sufficiently large proportion of the possible solutions. Most of the simpler or more appealing of these tests can be applied at reasonable computational cost essentially to only one constraint at a time. Two main approaches have been suggested for mitigating this limitation. One is to periodically apply linear programming to continuous approximations of the subproblems generated by the partial solutions. The other approach, promulgated by F. Glover, is to periodically introduce composite redundant constraints which tend to be useful when tests are applied to them individually. ~~In~~ this paper ~~we motivate a~~ measure of the "strength" of a composite constraint th~~at is slightly different from the one used by~~ Glover, and show how composite constraints that are as strong as possible in this sense can be computed by linear programming. It further develops that the dual of the required linear program coincides with the appropriate continuous approximation to the subproblems generated by the successive partial solutions. This leads to a complete synthesis of the two approaches mentioned above by means of an imbedded linear program. Computational experience is presented which confirms that this synthesis is indeed a useful one for the classes of problems tried. For numerous problems taken from the literature with up to 80 variables, the imbedded linear program typically reduced the number of required

iterations by one or two orders of magnitude, and execution times by a factor of between 3 and 20.

I. INTRODUCTION

The general 0-1 integer linear programming problem is:

(P) Minimize cx subject to $b + Ax \geq 0$, x_j binary,

where c and x are n-vectors, b is an m-vector, and A is an m x n matrix. The implicit enumeration approach to this problem has been the subject of considerable recent investigation (see, e.g., references 1, 2, 4, 7, 8, 10, 11). This approach is based on a "backtracking" procedure for what amounts to implicit enumeration of all 2^n possible solutions. Its efficiency depends on the ability to exclude a large proportion of the possible solutions from further consideration by means of various tests applied to partial solutions. A partial solution is a subset of the n variables with a specific binary value assigned to each. (The variables not in the subset are termed free.) The tests usually amount to examining the constraints in an effort to determine whether any completion of the current partial solution could possibly yield a feasible solution of (P) that has a lower value of the objective function than the best known feasible solution. Accordingly, the algorithm either continues by augmenting the current partial solution or backtracks to a different one.

Most of these tests can be applied at reasonable computational cost essentially to only one constraint at a time. Two main approaches have been suggested for mitigating this limitation. One is to periodically apply linear programming to the continuous verions of (P) in the free variables. The other approach, promulgated by Glover, [8] is to periodically introduce additional constraints which are redundant in the usual sense and yet effective when the tests are applied to them individually. We shall call these additional constraints _composite_ constraints, since they will be composed primarily (but not entirely) from the given constraints by non-negative linear combination.

In this paper we motivate a measure of the "strength" of a composite constraint that is slightly different from the one proposed by Glover. It then develops that strongest composite constraints can always be computed by linear programming, thereby obviating the need for approximate methods. It further develops that the dual of the required linear program coincides exactly with the continuous version of (P) in the free variables. This leads to a complete synthesis of the two approaches mentioned above. The available computational evidence suggests that this synthesis is indeed a useful one.

II. IMPLICIT ENUMERATION WITH COMPOSITE CONSTRAINTS

Denote a partial solution by an ordered set S, where each element is a non-zero integer between -n and n that may or may not be underlined. An element j (- j) of S indicates that x_j takes on the value 1 (0) in the partial solution. Using an obvious notation, we write $x_j^S = 1$ ($= 0$) if j (- j) is in S. The significance of an underline at the k^{th} position (counting from the left) is that all completions of the partial solution up to and including the k^{th} element complemented have been accounted for. Associated with any partial solution S is an integer program (P_S) involving the free variables (the variables not fixed by S):

$$\text{Minimize} \sum_{j \in S} c_j x_j^S + \sum_{j \notin S} c_j x_j \quad \text{subject to}$$

$$(P_S) \qquad b_i + \sum_{j \in S} a_{ij} x_j^S + \sum_{j \notin S} a_{ij} x_j \geq 0, \; i = 1, \ldots, m$$

$$x_j = 0 \text{ or } 1, \; j \notin S,$$

where the notation $j \in S$ ($\notin S$) refers to the fixed (free) variables.

In addition to the original m constraints, (P_S) may also be expanded to include one or more <u>composite</u> constraints, each of which is a non-negative linear combination of the original constraints plus the constraint $(\bar{z} - cx) > 0$, where $\bar{z}$ is the value of the currently best known feasible solution of (P).* More precisely, each composite

*If no feasible solution is known a priori (indeed, (P) may be infeasible), $\bar{z}$ can be initially taken as

$$\sum_{j=1}^{n} |c_j| .$$

constraint is the form

$$u(b + Ax) + (\bar{z} - cx) > 0$$

for some non-negative m-vector u. Such a constraint is clearly satisfied by any feasible solution of (P) that has a better value of cx than $\bar{z}$.

From the results of Ref. 7 or 8 it follows that the following procedure terminates in a finite number of steps either with an optimal solution of (P), or with an indication that no feasible solution of (P) exists with value less than the initial value of $\bar{z}$. The sequence of partial solutions generated is non-redundant in the appropriate sense.

A PROCEDURE FOR SOLVING (P) BY IMPLICIT ENUMERATION

Step 0: Initialize $\bar{z}$ at a known upper bound on the optimal value of (P), and S at an arbitrary partial solution without underlines.

Step 1: If (P_S) is obviously devoid of a feasible solution with value less than $\bar{z}$, go to Step 4. If (P_S) has an obvious optimal solution with value less than $\bar{z}$, then replace $\bar{z}$ by this value, store the optimal solution as the incumbent, and go to Step 4. If any free variable must obviously take on a particular binary value in order for (P_S) to have a feasible solution with value less than $\bar{z}$, then augment S on the right by $\underline{j}$ $(-\underline{j})$ for each variable x_j that must take on the value 1 (0).

Step 2: Add a new composite constraint and/or delete one or more current composite constraints, or do neither.

<u>Step 3</u>: Augment S on the right by $\pm j$ for some free variable (or several free variables) x_j.

<u>Step 4</u>: Locate the rightmost element of S which is not underlined. If none exists, terminate; otherwise, replace the element by its underlined complement and drop all elements to the right. Return to Step 1.

There is a wide variety of possible mechanisms for implementing Steps 1 and 3. Many can be found in, or adapted from, Refs. 1, 2, 4, 7, 8, 10 and 11. The possibilities are further multipled by the fact that the conditional instructions of Step 1 can be executed in any order or even in parallel. It is important to observe that many of the possible mechanisms, and perhaps most of the ones that are relatively inexpensive computationally, essentially apply to the constraints <u>only one at a time</u>. At Step 1, for example, a prominent role is often played by tests for binary-infeasibility and for conditional binary-infeasibility, with each constraint being considered individually. A constraint is said to be <u>binary-infeasible</u> if it has no binary solution, and is said to be <u>conditionally binary-infeasible</u> if its binary-feasibility is conditional upon certain of the variables taking on particular binary values. It is easily verified that $\beta + \Sigma_j \alpha_j x_j \geq 0$ (> 0) is binary infeasible if and only if $\beta + \Sigma_j \text{Max} \{0, \alpha_j\} < 0$ (≤ 0); and $\beta + \Sigma_j \text{Max} \{0, \alpha_j\} - |\alpha_{jo}| < 0$ (≤ 0) implies $x_{jo} = 0$ or 1 according as $\alpha_{jo} < 0$ or $\alpha_{jo} > 0$ in any binary solution satisfying $\beta + \Sigma_j \alpha_j x_j \geq 0$ (> 0).

This leads naturally to the desire to introduce composite constraints at Step 2 that are "strong" in the sense that such mechanisms are effective when applied to them.

III. COMPUTING COMPOSITE CONSTRAINTS

Since at any given stage of the calculations only a subset of the variables are free, the "strength" of a composite constraint must be defined relative to the current partial solution S. For simplicity we introduce the notation

$$z^S = \sum_{j \in S} c_j x_j^S \text{ and } b_i^S = b_i + \sum_{j \in S} a_{ij} x_j^S.$$

The special role played by conditional and unconditional binary-infeasibility (see Sec. II) suggests the following definition of the "strength" of a composite constraint.

Definition. The composite constraint $u^1(b + Ax) + (\bar{z} - cx) > 0$ is said to be stronger than the composite constraint $u^2(b + Ax) + (\bar{z} - cx) > 0$ relative to S if the maximum of the left-hand side of the first constraint is less than the maximum of the left-hand side of the second constraint, the maxima being taken over binary values of the free variables; i.e., if

$$\sum_{i=1}^{m} u_i^1 b_i^S + \bar{z} - z^S + \sum_{j \notin S} \text{Max } \{0, \sum_{i=1}^{m} u_i^1 a_{ij} - c_j\} < \sum_{i=1}^{m} u_i^2 b_i^S$$
$$+ \bar{z} - z^S + \sum_{j \notin S} \text{Max } \{0, \sum_{i=1}^{m} u_i^2 a_{ij} - c_j\}.$$

(For purposes of comparison, the corresponding definition used by Glover seems to be: the surrogate constraint $u^1(b + Ax) \geq 0$ is said to be stronger than the surrogate constraint $u^2(b + Ax) \geq 0$ relative to S if the maximum of $(\bar{z} - cx)$ subject to the first constraint is less than the maximum of $(\bar{z} - cx)$ subject to the second constraint, the maxima being taken over binary values of the free variables.)

Finding a strongest composite constraint is, then, the problem of minimizing

$$\sum_{i=1}^{m} u_i b_i^S - z^S + \sum_{j \notin S} \text{Max}\ \{0, \sum_{i=1}^{m} u_i a_{ij} - c_j\}$$

(we have dropped the constant $\bar{z}$) over all $u \geq 0$. But this problem is clearly equivalent to the following linear program:

$$(LP_S) \qquad \underset{u_i,\, w_j}{\text{Minimize}} \sum_{i=1}^{m} u_i b_i^S - z^S + \sum_{j \notin S} w_j \quad \text{subject to}$$

$$w_j \geq \sum_{i=1}^{m} u_i a_{ij} - c_j, \quad \text{all } j \notin S$$

$$u_i \geq 0, \quad i = 1, \ldots, m$$

$$w_j \geq 0, \quad i \notin S$$

Note that (LP_S) is necessarily feasible (for any choice of the u_i, let the w_j be sufficiently large). Denote the optimal value of (LP_S) by $v(LP_S)$. We thus have

<u>Theorem 1</u>: Let S be an arbitrary partial solution. Then (LP_S) is feasible, and

(i) $v(LP_S) = -\infty \Leftrightarrow$ there is no strongest composite constraint relative to S;

(ii) $v(LP_S) > -\infty \Rightarrow$ any optimal u yields a strongest composite constraint relative to S.

The usefulness of (LP_S) for finding strongest composite constraints is greatly enhanced by the fact that it is precisely the dual of $(\bar{P}_S)$, the continuous version of (P_S) (replace $x_j = 0$ or 1 by $0 \leq x_j \leq 1$). By the Dual Theorem of linear programming and the relationship between (P_S) and $(\bar{P}_S)$, one can easily prove

Theorem 2: Let S be an arbitrary partial solution. Then

(i) $v(LP_S) = -\infty \Leftrightarrow (\bar{P}_S)$ is infeasible $\Rightarrow (P_S)$ is infeasible.

(ii) $-\infty < v(LP_S) \leq -\bar{z} \Leftrightarrow (\bar{P}_S)$ is feasible and has optimal value $\geq \bar{z} \Rightarrow (P_S)$, if feasible, has optimal value $\geq \bar{z}$.

(iii) $v(LP_S) > -\bar{z} \Leftrightarrow (\bar{P}_S)$ is feasible and has optimal value $< \bar{z}$.

Furthermore, if $v(LP_S) > -\infty$ then the optimal dual variables of (LP_S) are optimal in (P_S) if they are integers.

The significance of this result is that it often enables the aim of Step 1 to be accomplished, at no extra computational cost, in the course of attempting to construct strongest composite constraints at Step 2. More specifically, one would set out to construct a strongest composite constraint by executing simplex iterations on (LP_S) until one of the following mutually exclusive events occurs: (a) the value of the objective function of (LP_S) becomes $\leq -\bar{z}$; (b) the optimal value of (LP_S) is reached and it is $> -\bar{z}$ and the optimal dual variables are all integers; (c) the optimal value of (LP_S) is reached and it is $> -\bar{z}$ and not all of the dual variables are integers. In event (a), a strong (binary infeasible, in fact) composite constraint is obtained from the values of the u_i variables in (LP_S), and one may go to Step 4; in event (b), the optimal solution of (P_S) is given by the optimal dual variables of (LP_S), so one should replace $\bar{z}$ by the new value and the incumbent by the new solution and go to Step 4; in event (c), a strongest composite constraint is obtained from the optimal u_i variables solving (LP_S).

Post-optimality techniques primal to (LP_S) can conveniently be used to take advantage of the results of previous calculations each time Step 2 is to be executed. (Since we do not always optimize (LP_S), some of the "optimality techniques" are "pre-" as well as "post-".) The Revised Simplex format is convenient for such techniques. Use should be made of the fact that the columns of the w_j are just the negatives of the unit vectors associated with the corresponding slack variables. One consequence is that the w_j can be treated logically rather than algebraically, so that (LP_S) is reduced to essentially m non-trival variables and as many constraints as free variables. The other important consequence is that it is easy to write down a basic feasible solution to (LP_S) for any S; in fact, there is an obvious and simple procedure for modifying a basic feasible solution for (LP_S) until it becomes basic feasible for $(LP_{S'})$, where $S' \neq S$. This avoids the need for post-optimality techniques that are dual to (LP_S).

IV. COMPUTATIONAL EXPERIENCE

The particular version of the implicit enumeration procedure chosen for implementation emphasizes simplicity of design and ease of programming above all. It is of completely general applicability, and takes no advantage of special problem structures. Step 1 uses just the simple tests for conditional and unconditional binary-infeasibility mentioned at the end of Sec. II; it recognizes an obvious optimal solution of (P_S) only by minimizing

$$\sum_{j \notin S} c_j x_j$$

over binary values of the free variables while ignoring the constraints, and then testing the resulting solution for feasibility. Step 2 follows the outline and suggestions given at the end of the previous section. Step 3 uses a simplified version of Balas' augmentation rule: Augment S by j_o, where j_o maximizes over all free variables the expression

$$\sum_{i=1}^{m} \text{Min} \{0, b_i^S + a_{ij}\}.$$

(This assumes, without loss of generality, that $c \geq 0$.)

The program was written entirely in Fortran IV for RAND's 32,000 word 7044. The object program and its data is all-in-core, treats all problem data as floating point, and will handle problems with up to 90 variables and 50 constraints (including composite constraints, if any). The linear programming subroutine is basically a Revised Simplex method with explicit inverse, the starting point having been a routine due to R. Clasen[3]. Pre/post-optimality techniques were incorporated that use a labeling procedure rather than more conventional matrix manipulations. The basis of the labeling procedure is the observation that fixing a variable at the value 0 or 1 can be viewed as demanding equality in the appropriate inequality constraint among $0 \leq x_j \leq 1$, $j \in S$, in the continuous version of (P_S). This means that the corresponding dual variables (the w_j and slacks in (LP_S)) become unconstrained in sign; the appropriate variables are therefore labeled and treated as "unsigned." This procedure, while easier to program than a more conventional one using matrix manipulations, has the drawback that (LP_S) (and therefore the explicit inverse) always has n rows, instead of only as many rows as free variables. Hence, each pivot requires more work, and additional core is used.

The code has been used to solve numerous different test problems with up to 80 variables taken from the literature (Refs. 1, 2, 4, 9, 10, 11 and 12). The number of iterations (executions of Step 1) and execution times (until termination, to the nearest hundredth of a minute) for most of these problems is presented in Table 1. We have omitted the problems too small to be of interest. Each problem was run twice: once skipping Step 2, so that no composite constraints were ever computed; and once with Step 2 fully implemented, so that an attempt was made to compute a new composite constraint each time, with only the last four composite constraints being kept and used. The columns corresponding to these runs are labeled "No LP" and "LP Every Time," respectively, in Table 1.

TABLE 1

PROBLEM DESIGNATION	PROBLEM SIZE 0-1 VAR x CONST.	NO LP[a] ITER.	NO LP[a] MIN.	LP EVERY TIME ITER.	LP EVERY TIME MIN.	OTHER ALGORITHMS MIN.	OTHER ALGORITHMS MACHINE	OTHER ALGORITHMS VERSION	OTHER ALGORITHMS REF.
B & M[2] 15	20 x 20	515	0.48	25	0.10	0.47	7044		2
16	20 x 20	1,897	1.69	39	0.62	2.07	7044		2
17	20 x 23	889	1.06	115	0.63	0.85	7044		2
18	20 x 23	569	0.59	1	0.04	0.97	7044		2
22	25 x 20	4,267	4.88	143	0.91	3.27	7044		2
23	27 x 20	6,565	8.08	171	1.16	7.10	7044		2
24	28 x 20	> 8,117	>10.00	281	1.91	15.20	7044		2
Fleischmann[4] I-35	35 x 8	1,009	0.50	17	0.04[b]	0.07[c]	7094		4
I-50	50 x 11		>10.00	25	0.04[b]	1.54[c]	7094		4
I-60	60 x 11	>10,367	>10.00	41	0.17	8.05	7094		4
I-80-1	80 x 11			35	0.21	~30	7094		4
I-80-2	80 x 11			45	0.32	~30	7094		4
L & S[10] B	35 x 28	1,995	2.39	63	0.25	5.5	360/40	DZIP1	10
C	44 x 12	2,061	1.90	451	1.37	5	360/40	DZIP1	10
D2	74 x 37	> 3,838	>10.00	>558	>10.00	>30	7094	DZIP1	10
Petersen[11] 3	15 x 10	159	0.06	71	0.04	1	SDS 930	REFORM.	11
4	20 x 10	609	0.31	81	0.06	4	SDS 930	REFORM.	11
5	28 x 10	5,013	3.45	101	0.08	46	SDS 930	REFORM.	11
"	"	"	"	"	"	3	SDS 930	R-1	11
6	39 x 5	>19,317	>10.00	297	0.54	6	SDS 930	R-1	11
7	50 x 5	>17,307	>10.00	423	1.12	>40	SDS 930	R-1	11
"	"	"	"	"	"	36	SDS 930	R-1 & R-2	11
Haldi[9] II-7	20 x 4	455	0.10	65	0.03	0.02	360/40	DZIP1	10
"	"	"	"	"	"	0.13	7090	LIP1	12
II-8	20 x 4	507	0.10	87	0.06	0.11	7090	LIP1	12
II-9	12 x 6	33	0.01	15	0.01	0.05	7090	LIP1	12
II-10	30 x 10	759	0.43	63	0.06	0.15	7090	LIP1	12
IBM[9] 1	21 x 7	435	0.14	17	0.01	0.03	7090	LIP1	12
2	21 x 7	369	0.11	27	0.03	0.05	7090	LIP1	12
3	20 x 3	87	0.02	9	0.01	0.05	7090	LIP1	12
4	30 x 15	>13,995	>10.00	77	0.19	0.19	7090	LIP1	12
"	"	"	"	"	"	6.67	7094	DZIP1	10
5	30 x 15	>13,858	>10.00	365	1.89	50	7094	DZIP1	10
"	"	"	"	"	"	1.11	7090	LIP1	12
6	31 x 31	> 7,038	>10.00	291	2.39	>50.00	7094	DZIP1	10
9	15 x 35	551	0.45	107	0.43	7.88	7090	LIP1	12

Table 1 Footnotes

[a]When termination did not occur within the 10-minute time limit, the best feasible solution yet found and the percent of the 2^n possible solutions that had been implicitly enumerated were printed out. For B & M 24, Fleischmann I-60, and IBM 5, an optimal solution was in store and the percent of the possible solutions accounted for was 47, 75, and 3 respectively. For L & S D2, no feasible solution was in store although 87.5 percent of the possible solutions were accounted for. For Petersen 6 and 7, and IBM 4 and 6, feasible solutions were available that were sub-optimal by 0.6 percent, 2.4 percent, 10 percent, 39 percent respectively, and the percent of possible solutions accounted for were 42.68, 0.77, 12.35 and 0.002 respectively.

[b]Average for five slightly different problems of the same size.

[c]Average for ten slightly different problems of the same size. In a recent communication[5], better times as a result of further modifications were announced as follows: for I-35, 0.04 min.; for I-50, 0.24 min.; for I-60, 1.68 min.; for I-80-1, 8.95 min.; for I-80-2, 8.28 min.

No prior information, such as an obvious initial feasible solution or upper bound on the optimal value of the objective function, was used. Such information was usually available, but we did not wish to further confound comparability with the computational results of other investigators, which are reproduced for easy reference in Table 1. These other investigators are Bouvier and Messoumian,[2] whose problems are randomly generated without any special structure at all; Fleischmann,[4] whose "economic" problems are highly structured; Lemke and Spielberg,[10] whose problems Band D2 were attributed to Mr. M. Sidrow of Texaco, and problem C to Mr. W. Acuri of IBM; Petersen,[11] whose problems are of a well-known capital budgeting variety; and Trauth and Woolsey,[12] who tested the LIP 1 code of Haldi and Isaacson among others on a number of problems including Haldi's fixed charge problems,[9] and some of the "IBM test problems" published by Haldi.[9] LIP 1 appears to be among the most efficient of the available codes based on Gomory's cutting-plane approach to linear integer programming. With this important exception, each of these investigators used a different adaptation of the implicit enumeration approach.

The data presented in Table 1 indicates that use of the imbedded linear program (LP_S) dramatically reduces the number of required iterations, typically by one or two orders of magnitude; and that this reduction is more than enough to pay for the time spent working on (LP_S), since execution times were typically reduced by a factor of between 3 and 20.

The present algorithm is evidently quite efficient relative to the others; but differences in programming and machine speed make it inadvisable to hazard a quantitative estimate of the apparent improvement.

No attempt has been made as yet to optimize program efficiency, or to try any of the many alternative implementations for Steps 1 and 3. For this reason the computational results of Table 1 should be considered as preliminary and subject to improvement. For example, the more powerful tests used by Fleischmann could be incorporated in the present program to improve its efficiency without the imbedded linear program, and therefore presumably with it. Significant reductions in computing time can also be achieved by computing composite constraints less often than at every opportunity. For example, Petersen 7 was solved in 0.60 instead of 1.12 minutes, and L&S C in 0.71 instead of 1.37 minutes, when (LP_S) was used every eighth time instead of every time. Another source of improvement would be the use of prior information.[7] As an illustration, inspection of the data for IBM 6 reveals an obvious good feasible solution, the use of which resulted in termination in .07 rather than in 2.39 minutes. Finally, we should point out that advantage could be taken of special problem structures. The tests introduced by Petersen in his modifications R-1 and R-2, for example, were very effective in taking advantage of the sign homogeneity in his capital budgeting problems.

REFERENCES

1. Balas, E., "An Additive Algorithm for Solving Linear Programs with Zero-One Variables," Operations Research, 13, 4 (July-August, 1965) 517-546.

2. Bouvier, B. and G. Messoumian, "Programmes Linéaires en Variables Bivalentes, Algorithme de Balas," Universitié de Grenoble, France, June, 1965.

3. Clasen, R. J., "Using Linear Programming as a Simplex Subroutine," The RAND Corporation, P-3267, November, 1965.

4. Fleischmann, B., "Computational Experience with the Algorithm of Balas," Operations Research, 15, 1 (January-February, 1967) 153-155.

5. Fleischmann, B., private communication, November 30, 1966.

6. Geoffrion, A. M., "An Improved Algorithm for Integer Programming by Implicit Enumeration," August 23, 1965, privately circulated.

7. Geoffrion, A. M., "Integer Programming by Implicit Enumeration and Balas' Method," The RAND Corporation, RM-4783-PR, to appear in SIAM Review.

8. Glover, F., "A Multiphase-Dual Algorithm for the Zero-One Integer Programming Problem," Operations Research, 13, 6 (November-December 1965), 879-919.

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

10. Lemke, C., and K. Spielberg, "Direct Search Zero-One and Mixed Integer Programming," June 1966, I.B.M. Technical Report 39.008, International Business Machines Corp., New York Scientific Center.

11. Petersen, C. C., "Computational Experience with Variants of the Balas Algorithm Applied to the Selection of R & D Projects," to appear in Management Science.

12. Trauth, C. A., and R. E. Woolsey, "Practical Aspects of Integer Linear Programming," Sandia Corporation Monograph SC-R-66-925, August 1966.

Unclassified

Security Classification

DOCUMENT CONTROL DATA - R&D

(Security classification of title body of abstract and indexing annotation must be entered when the overall report is classified)

1 ORIGINATING ACTIVITY (Corporate author) Western Management Science Institute, University of California, Los Angeles, California 90024	2a REPORT SECURITY CLASSIFICATION Unclassified
	2b GROUP
3 REPORT TITLE Implicit Enumeration Using An Imbedded Linear Program	
4 DESCRIPTIVE NOTES (Type of report and inclusive dates) Working Paper	
5 AUTHOR(S) (Last name, first name, initial) Geoffrion, Arthur M.	

6 REPORT DATE	7a TOTAL NO OF PAGES	7b NO OF REFS
June, 1967	16	12

8a CONTRACT OR GRANT NO Nonr 233(75)	9a ORIGINATOR'S REPORT NUMBER(S) Working Paper No. 120
b PROJECT NO	
c	9b OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
d	
10 AVAILABILITY/LIMITATION NOTICES Distribution of this document is unlimited.	Western Management Science Institute, University of California, Los Angeles, California 90024
11 SUPPLEMENTARY NOTES	12 SPONSORING MILITARY ACTIVITY

13 ABSTRACT Integer programming by implicit enumeration has been the subject of several recent investigations. Computational efficiency seems to depend primarily on the ability of various tests, applied to the constraints in connection with "partial solutions," to exclude from further consideration a sufficiently large proportion of the possible solutions. Most of the simpler or more appealing of these tests can be applied at reasonable computational cost essentially to only one constraint at a time. Two main approaches have been suggested for mitigating this limitation. One is to periodically apply linear programming to continuous approximations of the subproblems generated by the partial solutions. The other approach, promulgated by F. Glover, is to periodically introduce composite redundant constraints which tend to be useful when tests are applied to them individually. In this paper we motivate a measure of the "strength" of a composite constraint that is slightly different from the one used by Glover, and show how composite constraints that are as strong as possible in this sense can be computed by linear programming. It further develops that the dual of the required linear program coincides with the appropriate continuous approximation to the subproblems generated by the successive partial solutions. This leads to a complete systhesis of the two approaches mentioned above by means of an imbedded linear program. Computational experience is presented which confirms that this systhesis is indeed a useful one for the classes of problems tried. For numerous problems taken from the literature with up to 80 variables, the imbedded linear program typically reduced the number of required iterations by one or two orders of magnitude, and execution times by a factor of between 3 and 20.

DD FORM 1 JAN 64 1473 0101-807-6800

Unclassified

Security Classification

Security Classification

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Linear Programming Integer Programming Implicit Enumeration Computational Studies						

INSTRUCTIONS

1. ORIGINATING ACTIVITY: Enter the name and address of the contractor, subcontractor, grantee, Department of Defense activity or other organization *(corporate author)* issuing the report.

2a. REPORT SECURITY CLASSIFICATION: Enter the overall security classification of the report. Indicate whether "Restricted Data" is included. Marking is to be in accordance with appropriate security regulations.

2b. GROUP: Automatic downgrading is specified in DoD Directive 5200.10 and Armed Forces Industrial Manual. Enter the group number. Also, when applicable, show that optional markings have been used for Group 3 and Group 4 as authorized.

3. REPORT TITLE: Enter the complete report title in all capital letters. Titles in all cases should be unclassified. If a meaningful title cannot be selected without classification, show title classification in all capitals in parenthesis immediately following the title.

4. DESCRIPTIVE NOTES: If appropriate, enter the type of report, e.g., interim, progress, summary, annual, or final. Give the inclusive dates when a specific reporting period is covered.

5. AUTHOR(S): Enter the name(s) of author(s) as shown on or in the report. Enter last name, first name, middle initial. If military, show rank and branch of service. The name of the principal author is an absolute minimum requirement.

6. REPORT DATE: Enter the date of the report as day, month, year, or month, year. If more than one date appears on the report, use date of publication.

7a. TOTAL NUMBER OF PAGES: The total page count should follow normal pagination procedures, i.e., enter the number of pages containing information.

7b. NUMBER OF REFERENCES: Enter the total number of references cited in the report.

8a. CONTRACT OR GRANT NUMBER: If appropriate, enter the applicable number of the contract or grant under which the report was written.

8b, 8c, & 8d. PROJECT NUMBER: Enter the appropriate military department identification, such as project number, subproject number, system numbers, task number, etc.

9a. ORIGINATOR'S REPORT NUMBER(S): Enter the official report number by which the document will be identified and controlled by the originating activity. This number must be unique to this report.

9b. OTHER REPORT NUMBER(S): If the report has been assigned any other report numbers *(either by the originator or by the sponsor)*, also enter this number(s).

10. AVAILABILITY LIMITATION NOTICES: Enter any limitations on further dissemination of the report, other than those imposed by security classification, using standard statements such as:

(1) "Qualified requesters may obtain copies of this report from DDC."

(2) "Foreign announcement and dissemination of this report by DDC is not authorized."

(3) "U. S. Government agencies may obtain copies of this report directly from DDC. Other qualified DDC users shall request through ________________."

(4) "U. S. military agencies may obtain copies of this report directly from DDC. Other qualified users shall request through ________________."

(5) "All distribution of this report is controlled. Qualified DDC users shall request through ________________."

If the report has been furnished to the Office of Technical Services, Department of Commerce, for sale to the public, indicate this fact and enter the price, if known.

11. SUPPLEMENTARY NOTES: Use for additional explanatory notes.

12. SPONSORING MILITARY ACTIVITY: Enter the name of the departmental project office or laboratory sponsoring *(paying for)* the research and development. Include address.

13. ABSTRACT: Enter an abstract giving a brief and factual summary of the document indicative of the report, even though it may also appear elsewhere in the body of the technical report. If additional space is required, a continuation sheet shall be attached.

It is highly desirable that the abstract of classified reports be unclassified. Each paragraph of the abstract shall end with an indication of the military security classification of the information in the paragraph, represented as *(TS), (S), (C), or (U)*.

There is no limitation on the length of the abstract. However, the suggested length is from 150 to 225 words.

14. KEY WORDS: Key words are technically meaningful terms or short phrases that characterize a report and may be used as index entries for cataloging the report. Key words must be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location, may be used as key words but will be followed by an indication of technical context. The assignment of links, roles, and weights is optional.

Security Classification