Operations Research

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Technical Note—Generalized Covering Relaxation for 0-1 Programs

Daniel Granot, Frieda Granot,

To cite this article:
Daniel Granot, Frieda Granot, (1980) Technical Note—Generalized Covering Relaxation for 0-1 Programs. Operations Research 28(6):1442-1450. https://doi.org/10.1287/opre.28.6.1442

Generalized Covering Relaxation for 0-1 Programs

DANIEL GRANOT and FRIEDA GRANOT
University of British Columbia, Vancouver, Canada

(Received July 1979; accepted March 1980)

We construct in this paper a general purpose cutting-plane algorithm for solving the 0-1 polynomial programming problem of finding a 0-1 n vector $x = (x_j)$ that maximizes $c^T x$ subject to $f(x) \leq b$ where $f(x) = (f_i(x))$ is an m vector of polynomials. The algorithm consists of solving a nested sequence of linear generalized covering problems, i.e., covering problems involving both the original variables x and their complements $\bar{x} = 1 - x$. Each problem in the sequence is a relaxation of the original 0-1 polynomial program, and is obtained by adding to its predecessor a small number of generalized covering constraints that are violated by the optimal solution for the preceding generalized covering problem. Over 95% of more than 800 randomly generated problems with up to 70 variables and 50 constraints and mostly up to 5 terms in each constraint were solved by our method in less than 90 seconds of CPU time on an AMDAHL 470 V-6 computer.

IN THIS NOTE we consider a General 0-1 Polynomial programming (GP) problem of the form: find a 0-1 n-vector $x = (x_j)$ maximizing $c^T x$ subject to $f_i(x) = \sum_{k=1}^{P_i} a_{ik} \prod_{j \in N_{ik}} x_j \leq b_i$ for $i = 1, \cdots, m$, where $N_{ik} \subseteq N = \{1, \cdots, n\}$ and, without loss of generality, $c_j \geq 0$ for all j.

Nonlinear Integer Programming (NLIP) problems have numerous applications in economics and engineering, and arise when the decision variables in the nonlinear optimization models are restricted to integer values. See, for example, Soland (1973) for NLIP formulation of an optimal defensive missile allocation problem, and Zeitlin (1977) for various nonlinear integer resource allocation problems.

The importance of the 0-1 polynomial problem is magnified by the fact that every 0-1 Nonlinear Programming problem can be represented as an equivalent 0-1 polynomial problem in the same variables. Moreover polynomial terms in 0-1 variables have been found to be extremely useful in modeling in their own right; for example, they are used to incorporate risk into capital budgeting (Peterson and Laughhunn [1971]), for media selection problems (Zangwill [1965]), in cluster analysis (Rao [1971]), in various scheduling problems (Hammer and Rudeanu [1968], Pritsker et al. [1969]), for sewage treatment models (Oron [1975b]), and for irrigation system design (Oron [1975a]).

Operations Research
Vol. 28, No. 6, November–December 1980
0030-364X/80/2806-1442 $01.25

It was shown by Granot and Hammer (1971, 1975) for the general GP problem and by Balas and Jeroslow (1971) for the linear GP problem that every GP problem is equivalent to a special structured linear 0-1 problem—the Generalized Covering (GC) problem—of the form: find a 0-1 n-vector $\bar{x} \equiv (\bar{x}_j) = (1 - x_j)$ that minimizes $c^T\bar{x}$ subject to $A\bar{x}^\alpha \geq 1$ where $A = (a_{ij})$ is a 0-1 matrix, $\bar{x}^\alpha \equiv (\bar{x}_j^{\alpha_j})$ and $\bar{x}_j$ equals $\bar{x}_j^{\alpha_j}$ if $\alpha_j = 1$ and x_j if $\alpha_j = 0$. We will refer to constraints of GC as generalized covering constraints. The equivalence between a GP problem and the associated GC problem was used by Granot and Hammer (1971) to construct an algorithm for solving GP problems. Unfortunately, this attempt failed, mainly because of the large number of constraints in the equivalent GC problem.

Another approach for solving nonlinear GP problems is to first linearize the nonlinear constraints and then solve the equivalent linear problem; see, for example, Glover and Woolsey (1973, 1974) and Watters. However, a severe shortcoming of this approach is that it radically increases the dimension of the problem.

In Granot et al. (1979) an efficient general purpose cutting-plane algorithm was developed for solving the positive GP problem, that is, when the coefficients of the f_i's are all non-negative. In this paper we extend the results of Granot et al. to solve general GP problems. The algorithm solves a nested sequence of GC problems each of which is a relaxation of the original GP problem. Though the algorithm is applicable to both linear and nonlinear GP problems, its promise lies mainly for the nonlinear case. One of its main advantages is that no additional variables are introduced in the solution procedure.

Computational results for randomly generated GP problems are reported in Section 2.

1. AN ALGORITHM FOR SOLVING GP PROBLEMS

In this section we construct a general purpose algorithm for solving GP problems. We first generate a generalized covering problem that is a relaxation of GP, referred to as the Generalized Covering Relaxation (GCR) problem. If the optimal solution to GCR is feasible for GP, it is also optimal. Otherwise, GCR is augmented with additional generalized covering constraints that eliminate its current optimal solution. The number of constraints in each GCR problem solved is relatively small in comparison with the number of constraints in the equivalent GCR problem of Granot and Hammer (1971). Additional constraints are generated only to eliminate the optimal solution to GCR when not feasible for GP.

Let GCR be any relaxation of GP with $\hat{\bar{x}}$ an optimal solution of GCR. Assume that $\hat{x}$ is infeasible for GP. In the following we shall exhibit a method for generating a generalized covering constraint from a violated

polynomial constraint. The generalized covering constraint is a valid inequality for GP but is not satisfied by $\hat{x}$. For that purpose we introduce the following notation: let $f(x) = \sum_{k=1}^{P} a_k \prod_{J \in N_k} x_J \leq b$ be any constraint of GP that is violated by $\hat{x}$, and let $S^+ = \{k: a_k > 0 \text{ and } \prod_{J \in N_k} \hat{x}_J = 1\}$, $S^- = \{k: a_k < 0 \text{ and } \prod_{J \in N_k} \hat{x}_J = 1\}$, $S^0 = \{k: a_k < 0 \text{ and } \prod_{J \in N_k} \hat{x}_J = 0\}$, and $S = S^+ \cup S^0$. Furthermore, for each $k \in S^0$, let j_k be any element of N_k with $\hat{x}_{j_k} = 0$. Now for every 0-1 vector x we have

$$\begin{aligned} f(x) &\geq \sum_{k \in S^+} a_k \prod_{J \in N_k} x_J + \sum_{k \in S^0} a_k x_{j_k} + \sum_{k \in S^-} a_k \\ &= \sum_{k \in S^+} a_k \prod_{J \in N_k} x_J + \sum_{k \in S^0} (-a_k)\bar{x}_{j_k} + \sum_{k \in S^0 \cup S^-} a_k \equiv g(x). \end{aligned} \tag{1}$$

Denoting by $a_k' = |a_k|$, $y_k = \pi_{J \in N_k} x_J$ if $a_k > 0$ and $y_k = \bar{x}_{j_k}$ if $a_k < 0$, and substituting $b' = \sum_{k \in S^0 \cup S^-} a_k$ in (1) produces the valid constraint

$$g(x) = \sum_{k \in S} a_k' y_k + b' \leq b \tag{2}$$

for GP. Observe that (2) is violated by $\hat{x}$ since $g(\hat{x}) = f(\hat{x}) > b$.

Let $C \subseteq S$ be any subset of S for which $\sum_{k \in C} a_k' > b - b'$. Then $\pi_{k \in C} y_k = 0$ is a necessary condition for (2) to hold and is thus valid for GP. Substituting y in terms of x in $\pi_{k \in C} y_k = 0$ yields the equation $\pi_{k \in C \cap S^+}(\pi_{J \in N_k} x_J) \cdot \pi_{k \in C \cap S^0}(\bar{x}_{j_k}) = 0$ which, because x is 0-1, is equivalent to the generalized covering constraint

$$\sum_{J \in V} \bar{x}_J + \sum_{k \in C \cap S^0} x_{j_k} \geq 1 \tag{3}$$

where $V = \cup_{k \in C \cap S^+} N_k$. Now, (3) is a valid constraint for GP, but, since $\hat{y}_k = y_k(\hat{x}) = 1$ for each $k \in C$, it is violated by $\hat{x}$. Therefore, when (3) is added to GCR it cuts off the current optimal solution $\hat{x}$, but no other feasible solutions to GP.

One particular method to generate the set C, which is employed throughout our computational experiments, is to arrange the a_k''s in a decreasing order and then choose $C = \{1, \cdots, e\}$ where e is the smallest integer for which $\sum_{k=1}^{e} a_k' > b - b'$. Furthermore the index j_k was chosen to be the smallest index j in N_k for which $\hat{x}_j = 0$.

Example 1. Consider the following constraint: $f(x) = 5x_1x_2 - 3x_3x_4x_5 + 2x_2x_6 \leq 4$ and the solution $\hat{x} = (1, 1, 1, 1, 0, 0)$. Since $\hat{x}$ violates the constraint $f(x) \leq 4$ and $\hat{x}_5 = \hat{x}_6 = 0$ we generate a generalized covering constraint from $g(x) = 5x_1x_2 + 3\bar{x}_5 - 3 \leq 4$. The constraint generated is $\bar{x}_1 + \bar{x}_2 + x_5 \geq 1$.

We provide below a formal description of our algorithm for solving GP problems. The algorithm starts with the trivial GCR problem of finding a 0-1 vector x maximizing $c^T x$ whose optimal solution is $\hat{x} = 1$.

An Algorithm for Solving GP Problems

Step 1. Find an optimal solution $\hat{\hat{x}}$ for the GCR problem. If $\hat{x}$ is feasible

for GP, terminate with $\hat{x}$ an optimal solution thereof; otherwise go to Step 2.

Step 2. Generate at least one generalized covering constraint from each violated polynomial constraint at $\hat{x}$ and add them to the GCR problem forming a new GCR problem. Then go to Step 1.

Observe that every time Step 2 of the algorithm is applied, an optimal solution to GCR which is not feasible for GP is discarded. Since the number of binary vectors is 2^n, the finiteness of the algorithm follows. The optimality at termination stems from the fact that the GCR problem solved at Step 1 is a relaxation of GP.

Remark 1. Each one of the generalized covering constraints added to GCR in Step 2 is sufficient to eliminate $\hat{x}$. Adding more than one generalized cover will hopefully reduce the number of GCR problems needed to be solved. In the computational experiments performed, we generate one generalized covering constraint from each violated polynomial constraint of GP.

Remark 2. Note that no additional variables are introduced by the algorithm in the solution process and no special devices are needed to handle the nonlinear constraints.

2. COMPUTATIONAL RESULTS

Our algorithm was coded in Fortran IV and implemented on an AMDAHL 470 V-6 Computer model II. The main purpose of the computational experiments was to investigate the number of GCR problems (as well as their size) that need be solved in order to obtain an optimal solution to a GP problem. The algorithm was tested on more than 800 randomly generated problems with up to a maximum of 13 polynomial terms in each constraint. Ninety-five percent of these randomly generated problems were solved in 90 seconds or less of CPU time. Only 30% of the problems solved were found to be feasible. The problems were generated as follows. The cost vector c was determined by setting $c_0 = 0$ and $c_{j+1} = c_j + \Delta$ where Δ is uniform on [0, 10]. Every constraint is of the form $\sum_{j=1}^{p} a_{ij} y_j \leq b_i$ with $y_j = \pi_{t=1}^{e} x_{R_t}$ and is generated as follows: the number of terms p is uniform on [1, P], the number of variables e in each term is uniform on [3, 5], a_{ij} is uniform on [−20, 20], R_t is uniformly chosen between 1 and n, α is a constant between 0 and 1 that measures the tightness of the constraints, and the right hand side b_i is chosen to equal $\sum_{j=1}^{P} a_{ij} + \alpha \sum_{j=1}^{P} |a_{ij}|$. The results obtained from 792 problems that were run—11 for each combination of n, m and α—are summarized in Table I. The number of covers reported is the average number of generalized covering constraints in the last GCR problem solved. The number of problems solved designates the number of problems (out of 11) for which either an optimal solution was found or infeasibility was detected within

TABLE I

$P = 5$

No. of Variables	No. of Constraints	Tightness (α)	CPU Time (sec)	No. of Iterations	No. of Covers	No. of Feasible Problems	No. of Problems Solved Out of 11
50	25	0.90	0.09	3.3	8.9	11	11
50	30	0.90	0.24	4.2	11.3	11	11
50	35	0.90	0.58	6.2	13.3	10	11
60	25	0.90	0.05	3.5	8.9	11	11
60	30	0.90	0.15	3.7	10.4	11	11
60	35	0.90	0.54	5.3	13.1	10	11
70	25	0.90	0.06	3.2	8.5	11	11
70	30	0.90	0.12	3.9	10.4	11	11
70	35	0.90	0.45	5.5	13.2	10	11
50	25	0.70	0.77	7.4	16.6	10	10
50	30	0.70	1.32	8.3	20.0	8	10
50	35	0.70	6.85	7.0	23.1	6	10
60	25	0.70	0.96	5.0	15.6	11	11
60	30	0.70	1.71	6.4	18.5	10	10
60	35	0.70	7.27	7.5	24.0	7	10
70	25	0.70	2.63	5.1	16.0	11	11
70	30	0.70	11.09	8.0	21.8	10	11
70	35	0.70	11.38	7.2	24.5	7	10
50	25	0.50	1.93	5.9	24.5	4	11
50	30	0.50	5.68	4.7	26.5	0	10
50	35	0.50	3.51	3.0	28.0	0	11
60	25	0.50	9.74	5.5	24.7	6	11
60	30	0.50	1.66	5.1	27.8	1	9
60	35	0.50	1.93	3.7	28.0	0	11
70	25	0.50	9.46	6.3	22.0	9	10
70	30	0.50	8.99	5.9	27.4	5	11
70	35	0.50	10.59	4.9	30.7	2	10
50	25	0.40	1.26	2.8	23.9	0	11
50	30	0.40	1.14	2.1	26.5	0	11
50	35	0.40	2.48	1.6	29.2	0	11
60	25	0.40	1.42	2.9	23.6	1	11
60	30	0.40	1.16	2.2	26.3	0	11
60	35	0.40	3.73	1.9	29.7	0	11
70	25	0.40	6.04	3.9	23.9	1	11
70	30	0.40	4.62	2.6	26.3	1	11
70	35	0.40	4.73	2.1	29.9	0	10
50	25	0.30	0.47	1.6	24.7	0	11
50	30	0.30	0.95	1.5	28.2	0	11
50	35	0.30	1.24	1.4	31.5	0	11

TABLE I—*Continued*

No. of Variables	No. of Constraints	Tightness (α)	CPU Time (sec)	No. of Iterations	No. of Covers	No. of Feasible Problems	No. of Problems Solved Out of 11
60	25	0.30	1.76	1.9	25.9	0	11
60	30	0.30	1.67	1.8	29.1	0	11
60	35	0.30	4.74	1.5	33.2	0	11
70	25	0.30	5.55	2.4	26.5	0	11
70	30	0.30	8.02	1.9	29.1	0	11
70	35	0.30	8.33	1.5	33.2	0	10
50	25	0.25	0.93	1.4	26.6	0	11
50	30	0.25	1.43	1.2	30.6	0	11
50	35	0.25	3.55	1.1	35.1	0	11
60	25	0.25	1.64	1.3	27.3	0	11
60	30	0.25	2.58	1.3	31.3	0	11
60	35	0.25	9.44	1.2	35.7	0	11
70	25	0.25	3.88	1.8	27.3	0	11
70	30	0.25	6.00	1.6	31.5	0	11
70	35	0.25	27.17	1.5	35.9	0	11
50	25	0.20	1.93	1.2	29.4	0	11
50	30	0.20	2.86	1.0	34.0	0	11
50	35	0.20	7.49	1.0	39.1	0	11
60	25	0.20	2.60	1.0	29.5	0	11
60	30	0.20	6.53	1.0	34.3	0	11
60	35	0.20	20.67	1.0	39.6	0	11
70	25	0.20	11.00	1.5	29.9	0	11
70	30	0.20	10.29	1.3	34.4	0	11
70	35	0.20	26.48	1.1	39.0	0	10
50	25	0.10	5.99	1.1	34.5	0	11
50	30	0.10	6.67	1.0	40.5	0	11
50	35	0.10	14.81	1.0	46.9	0	11
60	25	0.10	12.30	1.0	34.8	0	11
60	30	0.10	21.01	1.1	40.7	0	11
60	35	0.10	47.63	1.0	47.3	0	11
70	25	0.10	22.33	1.1	34.7	0	11
70	30	0.10	23.10	1.0	40.5	0	11
70	35	0.10	43.06	1.0	45.1	0	3

90 seconds of CPU time. The CPU time reported is the average of the true execution times in seconds for those problems solved. We remark that about 90% of the computational time was spent in solving the GCR problems. Those problems were solved using the implicit enumeration method of Balas (1965). Neither surrogate constraints nor subgradient optimization techniques were employed; however, various simplifications and elimination rules were developed and applied to the coefficient matrix

of each GCR problem before solving it by the implicit enumeration method.

In Table II the number of problems solved designates the number of problems (out of 6) for which either an optimal solution was found or infeasibility was detected in 60 seconds of CPU time.

The following conclusions can be drawn from Tables I and II.

1. An increase in the number of variables significantly increases the computational time, but has only a very modest effect on both the number of iterations and the average size of the generalized covering problems solved.

2. The tightness of the constraints, as measured by the parameter α, has a significant effect on computational performance. Both the number of iterations and the average size of the covering problems solved are unimodal functions of α with the maximum attained at $\alpha = 0.7$ and $\alpha =$

TABLE II

$\alpha = 0.50$

	10 Constraints × 30 Variables					20 Constraints × 40 Variables				
Maximum No. of Terms (P)	CPU time (sec)	No. of iterations	No of covers	No. of problems solved out of 6	No of feasible problems	CPU time (sec)	No. of iterations	No. of covers	No of problems solved out of 6	No. of feasible problems
3	0.04	3.2	8.2	6	5	0.32	4.2	13.2	6	4
5	0.11	5.4	10.3	6	6	2.19	6.3	22.4	6	2
7	0 13	6.3	12.6	6	6	4 68	9 6	29.5	6	3
9	0.36	9.1	12.5	6	5	11.07	10.9	45.7	6	1
11	0.94	9.7	29.3	6	6	22.88	12.6	56.3	6	2
13	4.39	20.4	49.6	6	6	26 31	19 8	81 2	3	3[a]

[a] The three problems that were solved were all feasible.

0.5, respectively. However, the computational time as a function of α is bimodal, where the two most difficult ranges are $\{0.5 \leq \alpha \leq 0.7\}$ and $\{\alpha \leq 0.2\}$. This last feature did not occur in our covering relaxation algorithm for solving positive 0-1 polynomial programs, where the computational time is a unimodal function of α attaining its maximum for $\{0.4 \leq \alpha \leq 0.7\}$ (Granot et al. [1979]).

3. An increase in the number of constraints significantly influences the computational performance. In fact an increase in m seems to have a similar effect on computational performance as a decrease in α.

4. Table II reveals the significance of an increase in the number of terms in each constraint on computational results. An increase in P significantly increases the CPU time, the number of iterations and the size of the generalized covering problems solved. In fact, three out of six 20 constraints and 40 variables GP problems with $P = 13$ require more than 1 minute of CPU time.

ACKNOWLEDGMENTS

Research and reproduction of this report were partially supported by the Air Force office of Scientific Research Contract F 44620-74-C-0079. Research was also supported by NRC grant A4181 and NRC grant A3998. This report was also issued as Report Number 79 under the Office of Naval Research Contract N00014-76-C-0418.

We are grateful to Mr. W. Vaessen for coding the algorithm and performing the computational experiments. The authors are also thankful to Professor A. F. Veinott, Jr., for his constructive remarks made on an earlier version of this paper.

REFERENCES

BALAS, E. 1965. An Additive Algorithm for Solving Linear Programs with 0-1 Variables. *Opns. Res.* **13,** 517–546.

BALAS, E., AND R. JEROSLOW. 1969. Canonical Cuts on the Unit Hypercube. Management Science Research Report No. 196(R), Carnegie-Mellon University, August-December. Revised February 1971.

GLOVER, F., AND E. WOOLSEY. 1973. Further Reduction of Zero-One Polynomial Programming Problems to Zero-One Linear Programming Problems. *Opns. Res.* **21,** 141–161.

GLOVER, E., AND E. WOOLSEY. 1974. Converting the 0-1 Polynomial Programming Problem to a 0-1 Linear Program. *Opns. Res.* **22,** 180–182.

GRANOT, D., F. GRANOT AND J. KALLBERG. 1979. Covering Relaxation for Positive 0-1 Polynomial Programs. *Mgmt. Sci.* **25,** 264–273.

GRANOT, F., AND P. L. HAMMER. 1971. On the Use of Boolean Functions in 0-1 Programming. *Methods Opns. Res.* **12,** 154–184.

GRANOT, F., AND P. L. HAMMER. 1975. On the Role of Generalized Covering Problems. *Cahiers Cent. Etudes Rech. Operationelle* **17,** 277–289.

HAMMER, P. L., AND S. RUDEANU. 1968. *Boolean Methods in Operations Research.* Springer-Verlag, New York.

ORON, G. 1975a. Optimizing Irrigation System Design, Ph.D. dissertation Agriculture Engineering Department, Technion, Haifa, Israel.

ORON, G. 1975b. An Algorithm for Optimizing Nonlinear Constrained Zero-One Problems. Technical Report, Colorado State University, Fort Collins.

PETERSON, D. E., AND D. LAUGHHUNN. 1971. Capital Expenditure Programming and Some Alternative Approaches to Risk. *Mgmt. Sci.* **17,** 320–336.

PRITSKER, A., L. J. WATTERS AND F. WOLFE. 1969. Multiproject Scheduling with Limited Resources: A Zero-One Programming Approach. *Mgmt. Sci.* **16,** 93–108.

RAO, M. R. 1971. Cluster Analysis and Mathematical Programming. *J. Am. Statist. Assoc.* **66,** 622–626.

SOLAND, R. M. 1973. Optimal Defensive Missile Allocation: A Discrete Min-Max Problem. *Opns. Res.* **21,** 590–596.

WATTERS, L. J. 1967. Reduction of Integer Polynomial Programming Problems to Zero-One Linear Programming Problems. *Opns. Res.* **15,** 1171–1174.

ZANGWILL, W. 1965. Media Selection by Decision Programming. *J. Advertising Res.* **5**, 30–36.

ZEITLIN, Z. 1977. Some Nonlinear Programming Problems with Integer Resource Allocations. Ph.D. dissertation, Tel-Aviv University, September.

Improved Bounds for Aggregated Linear Programs

ROY MENDELSSOHN

University of California, Santa Cruz, California

(Received September 1978; accepted April 1980)

A method of Kallio for improving bounds on the optimal value of a linear program calculated from an intermediate iteration is used to improve Zipkin's bounds for an aggregated linear program. Both theoretical and computational results are given, demonstrating the improvement due to these new bounds.

ONE METHOD for solving large scale linear programs (LPs) or large scale Markov decision processes (MDPs) is by first finding an exact solution to a smaller, aggregated problem; disaggregating this solution to the original problem; and finding bounds on the error from using the approximate solution. This has been studied extensively by Zipkin (1980) for fixed-weight aggregation.

In related work, Kallio (1977) derives bounds for a nonaggregated LP which is stopped at some iteration of the simplex method, and uses a decomposition technique and marginal analysis to tighten the bounds. For nonaggregated LPs, Kallio's Theorem 1 and Zipkin's Proposition 2 are identical. In this paper, Kallio's method is used to improve the bounds for aggregated LPs and for fixed-weight aggregated MDPs. The notation of this paper follows that of Zipkin whenever possible.

1. The Model

The original LP is

$$\begin{aligned} z^* = \max\ & cx \\ \text{subject to } & Ax \leq b \\ & x \geq 0 \end{aligned} \tag{1}$$

where $c = (c_j)$ is an n-vector, $b = (b_i)$ is an m-vector, $A = (a_{ij})$ is an $m \times n$ matrix, and $x = (x_j)$ is an n-vector of variables.

Let $\sigma = \{S_k : k = 1, \cdots, K\}$ be an arbitrary partition of $\{1, \cdots, n\}$, and

Operations Research
Vol. 28, No. 6, November–December 1980

0030-364X/80/2806-1450 $01.25