Operations Research

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

Technical Note—Complementary Programming

Toshihide Ibaraki,

To cite this article:
Toshihide Ibaraki, (1971) Technical Note—Complementary Programming. Operations Research 19(6):1523-1529. https://doi.org/10.1287/opre.19.6.1523

COMPLEMENTARY PROGRAMMING

Toshihide Ibaraki

Kyoto University, Kyoto, Japan

(Received August 25, 1969)

A mathematical programming problem P: minimize $z=d^Tx+e^Tu+f^Tv$ subject to $Ax+Bu+Cv\geqq g$, x, u, $v\geqq 0$, $u^Tv=0$, is called a complementary programming (CP) problem. The condition $u^Tv=0$ distinguishes CP problems from ordinary LP problems. A variety of nonlinear programming problems can be formulated in this form, including absolute-value programming problems, 0–1 mixed-integer-programming problems, quadratic-programming problems, and so forth. This paper proposes a branch-and-bound algorithm for CP problems and reports some computational results.

THIS PAPER considers a complementary programming (CP) problem defined by

$$P\text{: minimize } z=d^Tx+e^Tu+f^Tv \text{ subject to } Ax+Bu+Cv\geqq g,\ x,\ u,\ v\geqq 0,\ u^Tv=0,$$

where: x, u, v are, respectively, n-, m-, m-vectors of variables; d, e, f are n-, m-, m-vectors of constants; A, B, C are $p\times n$, $p\times m$, $p\times m$ matrices of constants; and g is an m-vector of constants. All vectors are assumed to be column vectors, unless transposed (denoted by a superscript T).

The condition $u^Tv=0$ distinguishes CP problems from ordinary LP problems, since at least one of u_j and v_j for each j is forced to 0.

A special form of P disregarding the objective function $u-Mv=g$, $u^Tv=0$, where M is a $p\times p$ matrix, has received a certain amount of attention.[2, 9] The problem P, however, can include a much wider class of problems as discussed in the next section.

APPLICATIONS

TWO TYPICAL examples of CP problems will be presented. It is noted here that, in addition to them, other important programming problems, such as quadratic programming problems and LP problems with either-or conditions, can also be formulated as CP problems.

1. *A 0–1 mixed-integer-programming problem.* A 0–1 mixed-integer-programming problem is written as follows:

$$\text{minimize } d^Tx+e^Tu, \text{ subject to } Ax+Bu\geqq g,\ x\geqq 0,\ u_j=0 \text{ or } 1,\ j=1, 2, \cdots, m.$$

By introducing a slack variable v_j for each j, the 0–1 condition can be written $u_j+v_j=1$, $u_jv_j=0$, $j=1, 2, \cdots, m$. Thus this problem is equivalent to the following CP problem:

$$\text{minimize } d^Tx+e^Tu \text{ subject to } Ax+Bu\geqq g,\ x,\ u,\ v\geqq 0,\ u^Tv=0,\ u+v=\begin{bmatrix}1\\1\\\vdots\\1\end{bmatrix}.$$

2. *An absolute-value programming problem.* Consider an absolute-value programming problem:

$$\text{minimize } d^Tx+e^T|x|, \text{ subject to } Ax+B|x|\geqq g,$$

where $|x|=(|x_1|, |x_2|, \cdots, |x_n|)$. By putting $x_j=u_j-v_j$, $j=1, 2, \cdots, n$, we can represent $|x_j|$ by $|x_j|=u_j+v_j$, $u_jv_j=0$. Thus, this problem is equivalent to the following CP problem:

$$\text{minimize } d^T(u-v)+e^T(u+v), \text{ subject to } A(u-v)+B(u+v)\geqq g,\ u,\ v\geqq 0,\ u^Tv=0.$$

SOME DEFINITIONS

THE LATER sections give a branch-and-bound algorithm for solving the CP problem P. Here we state several definitions necessary for describing it.

For problem P, a point (x, u, v) that satisfies the constraints $Ax+Bu+Cv\geqq g$, $x,\ u,\ v\geqq 0$, $u^Tv=0$ is a *feasible solution*. A feasible solution that minimizes z is an *optimal solution*.

A *partial solution* S is an assignment of 0's to a subset of variables u and v in which both u_j and v_j never appear simultaneously. We adopt the notational convention that j denotes $u_j=0$ and $-j$ denotes $v_j=0$. For example, $S=\{-4, 1, -2\}$ implies that $v_4=0$, $u_1=0$, and $v_2=0$. Any variables u_j and v_j such that neither j nor $-j$ is included in S are *free*. If a partial solution T such that $T\supset S$ has no free variable, T is a *completion* of S.

With each partial solution S, the *partial problem* $P(S)$ is associated, where $P(S)$ is defined by

$$P(S)\text{: minimize } z=d^Tx+e^Tu+f^Tv, \text{ subject to}$$

$$Ax+Bu+Cv\geqq g,\ x,\ u,\ v\geqq 0,\ u^Tv=0,\ u_j=0 \text{ if } j\in S,\ v_j=0 \text{ if } -j\in S.$$

The value of $P(S)$, i.e., the value of z for the optimal solution to $P(S)$, is denoted by $z^0(S)$.

The LP problem obtained by deleting the condition $u^Tv=0$ from P is denoted by $\bar{P}$. Similarly, $\bar{P}(S)$ is $P(S)$ with the condition $u^Tv=0$ deleted. $\bar{z}(S)$ stands for the value of $\bar{P}(S)$.

THE BRANCH-AND-BOUND ALGORITHM FOR THE CP PROBLEM

THE BRANCH-AND-BOUND algorithm proposed here consists of a sequence of systematic decompositions of partial problems into smaller ones, followed by attempts to solve them. At the outset, only one problem $P(\phi)$ $(=P)$ is generated. As the computation proceeds, partial problems $P(S)$ are successively decomposed into $P(S, j)$ and $P(S, -j)$ by fixing appropriate variables u_j and v_j to 0, respectively.

From time to time, feasible solutions to P may be obtained as optimal solutions

to some partial problems generated by them. The best (minimum objective value) among them is maintained as the *incumbent*. The objective value of the incumbent is denoted by z^*.

If it is proved that a partial problem $P(S)$ is no longer useful to generate an optimal solution to P, $P(S)$ is said to be *terminated*. Any $P(S)$ neither terminated nor decomposed into smaller ones is *active*. When there exists no active partial problem, the computation terminates, possessing an optimal solution to P as the incumbent.

The whole procedure may be summarized as follows:

Step 1. Start with one active partial problem $P(\phi)$.

Step 2. Select an active partial problem $P(S)$. If none exists, terminate the computation.

Step 3. (i) If an optimal solution to $P(S)$ is obtained, terminate $P(S)$, and go to Step 4.

(ii) If it is proved that $P(S)$ has no feasible completion or that $P(S)$ cannot yield any feasible solution better than the incumbent, terminate $P(S)$ and return to Step 2.

(iii) If it is proved that $u_j=0$ must hold if feasible completions of $P(S)$ better than the incumbent are to be obtained, then return to Step 2 after generating the active partial problem $P(S, j)$ and terminating $P(S, -j)$. Similarly for the $v_j=0$ case.

(iv) If none of (i), (ii), and (iii) is satisfied, go to Step 5.

Step 4. Keep the optimal solution to $P(S)$ as the new incumbent if it is better than the old incumbent. Terminate all the active partial problems for which it is proved that they have no feasible completions better than the new incumbent. Return to Step 2.

Step 5. Select a pair of free variables u_j and v_j (*branching variables*). Generate two active partial problems $P(S, j)$ and $P(S, -j)$ from $P(S)$. Return to Step 2.

This is a straightforward adaptation of the general branch-and-bound principle[4,8] to the CP problem. The details of each step (Steps 2–5), however, need to be specified for the algorithm to be implemented. They will be discussed in the following sections.

SELECTION OF AN ACTIVE PARTIAL PROBLEM

As THE SELECTION rule for an active partial problem in Step 2, the rule that selects the active partial problem most recently generated (sometimes called the linear search[1]) is employed. Among a variety of selection schemes,[8] this is chosen because less storage space is required.

PENALTY FOR EACH FREE VARIABLE

THIS SECTION defines the penalty for each free variable, based on the manipulation of a simplex tableau for the LP problem $\bar{P}(S)$. This penalty, combined with the optimal solution to $\bar{P}(S)$, is quite useful in implementing Steps 3–5 in the branch-and-bound algorithm.

Assume that $\bar{P}(S)$ is solved and that z and $u_j(u_j>0)$ are expanded as follows in terms of nonbasic variables $t_1, t_2, \cdots, t_N(N=n+2m)$ in the final tableau:

$$z=\alpha_{00}+\sum_{k=1}^{k=N}\alpha_{0k}t_k,\ u_j=\alpha_{r0}+\sum_{k=1}^{k=N}\alpha_{rk}t_k,\ (\alpha_{r0}>0).$$

Let us define the penalty for u_j by

$$p(u_j)=\begin{cases}\infty, \text{ if } \alpha_{rk}\geqq 0,\ k=1, 2, \cdots, N,\\ -\alpha_{r0}\alpha_{0s}/\alpha_{rk}(\geqq 0), \text{ otherwise,}\end{cases}$$

where

$$\alpha_{0s}/\alpha_{rs}=\max_k\{\alpha_{0k}/\alpha_{rk}|\alpha_{rk}<0\}.$$

This $p(u_j)$ is a lower bound of the increase of z when u_j is fixed to 0, since this is the increase in z after the first pivot operation for driving u_j out of basis (note that, after the first pivot operation, u_j is expressed as $u_j=0+u_j$) while keeping the dual feasibility. Thus, the following relation results:

$$\bar{z}(S, j)\geqq\bar{z}(S)+p(u_j).$$

$p(v_j)$ is defined similarly. (This argument is similar to Driebeek's[3] for 0–1 problems.)

IMPLEMENTATION OF STEP 3 AND STEP 4

FROM THE FACT that $\bar{P}(S)$ is less constrained than $P(S)$, i.e., $z^0(S)\geqq\bar{z}(S)$, and the properties of penalties discussed in the previous section, it is now clear that the following properties can be used in implementing Step 3 and Step 4 of the branch-and-bound algorithm:

1. If an optimal solution to $\bar{P}(S)$ satisfies the condition $u^Tv=0$, then it is also optimal for $P(S)$. [Step 3(i).]

2. If $\bar{P}(S)$ is infeasible, so is $P(S)$. [Step 3(ii).]

3. If $\bar{z}(S)\geqq z^*$ (the value of the incumbent), then $P(S)$ has no feasible completion better than the incumbent. [Step 3(ii) and Step 4.]

4. If $\bar{z}(S)+p(u_j)\geqq z^*$, then partial problem $P(S, j)$ can be terminated. Similarly, if $\bar{z}(S)+p(v_j)\geqq z^*$, $P(S, -j)$ can be terminated. [Step 3(iii).]

SELECTION OF BRANCHING VARIABLES

TO IMPLEMENT Step 5 of the algorithm, the following method, based on penalties, is used.

Let u_s and v_s satisfy

$$|p(u_s)-p(v_s)|=\max_{j:\text{free},\ u_jv_j>0}|p(u_j)-p(v_j)|;$$

then the branching variables are u_s and v_s. With this s, $P(S, s)$ is selected first in Step 2 if $p(u_j)\leqq p(v_j)$; otherwise, $P(S, -s)$ is first.

This rule is employed because u_s and v_s defined as above tend to maximize the difference $|\bar{z}(S, j)-\bar{z}(S, -j)|$ among all free variables u_j and v_j. It is commonly noticed in the branch-and-bound method that this sort of approach is effective in reducing the number of partial problems generated in the computation.

TABLE I

COMPUTATIONAL RESULTS

Problem	n	m	$n+2m$[a]	p[b]	(c)	Time[d]	Other results[e] Time (secs)	Other results[e] Machine
R and D problem[f]	0	10	20	20	31	1.41		
"	0	20	40	30	59	7.47	2.4[5]	7044
"	0	28	56	38	143	37.12	14.4	7044
Switching network design[g]	17	4	25	28	10	1.67		
Fixed-charge problem[h] 7	3	2	7	9	6	0.09	1.8[5]	7044
" 8	3	2	7	9	5	0.08	3.0	7044
" 10	6	6	18	22	7	0.78	3.6	7044
Graph problem[i]	0	10	20	30	20	1.41	3.38[12]	CDC3600

TABLE II

THE EFFECT OF THE RATIO OF COMPLEMENTARY VARIABLES

Problem[j]	n	m	p[b]	(c)	Time in seconds[d]
1[k]	18	2	30	3.5	0.90
2	16	4	30	5.5	1.00
3	14	6	30	12.0	1.60
4	12	8	30	18.5	1.90
5	10	10	30	20.5	2.26
6	8	12	30	24.5	2.35
7	6	14	30	40.5	3.73
8	4	16	30	48.0	4.41
9	2	18	30	73.0	6.99
10	0	20	30	96.0	8.06

Notes for Tables I and II

[a] $n + 2m$ is the total number of variables.

[b] p is the number of constraints excluding $u^T v = 0$.

[c] Iterations: the number of partial problems actually tested.

[d] Computation time in seconds. The machine is FACOM 230/60.

[e] These computational results are obtained as 0-1 all integer problems.

[f] Taken from PETERSEN.[11]

[g] Taken from MUROGA AND IBARAKI;[10] the design of the parity function of two variables with threshold gates.

[h] Taken from HALDI.[6]

[i] Taken from TRAUTH AND WOOLSEY;[12] the combinational problem for a graph with 5 nodes.

[j] The original problem is the R and D selection problem[11] with 20 0-1 variables and 10 constraints. It is transformed into CP problems after randomly choosing m out of 20 as 0-1 variables and leaving the rest as continuous variables.

[k] The result in each row is the average of two problems.

COMPUTATIONAL RESULTS

THE ALGORITHM is coded in FORTRAN, using the properties given in the preceding four sections, and run on the FACOM 230/60 of Kyoto University. The machine corresponds very roughly to the IBM 360/65 or the UNIVAC 1108.

The computational results are shown in Tables I and II. Table I shows the results for the CP problems converted from 0–1 (mixed) integer problems (taken from the literature) following method 1 of the second section. Thus each problem has a larger number of constraints than the original.

Table II shows the effect of the number of variables restricted to be $u_j v_j = 0$. The computation time appears to increase exponentially as the number of variables restricted to be $u_j v_j = 0$ grows.

Other computational results and a listing of the FORTRAN program are available through the author.

ACKNOWLEDGMENT

THE AUTHOR wishes to express his appreciation to H. MINE of Kyoto University for his comments on the subject.

REFERENCES

1. M. L. BALINSKI AND K. SPIELBERG, "Methods for Integer Programming: Algebraic, Combinatorial, and Enumerative," in *Progress in Operations Research III*, J. S. ARONOFSKY (ed.), Wiley, New York, 1969.
2. R. W. COTTLE AND G. B. DANTZIG, "Complementary Pivot Theory of Mathematical Programming," *Linear Algebra and Its Applications* **1**, 103–125 (1968).
3. N. J. DRIEBEEK, "An Algorithm for the Solution of Mixed Integer Programming Problems," *Management Sci.* **12**, 576–587 (1966).
4. A. M. GEOFFRION, "Integer Programming by Implicit Enumeration and Balas' Method," *SIAM Review* **9**, 178–190 (1967).
5. ———, "An Improved Implicit Enumeration Approach for Integer Programming," *Opns. Res.* **17**, 437–454 (1969).
6. J. HALDI, "25 Integer Programming Test Problems," Working Paper, No. 43, Graduate School of Business, Stanford Univ., Stanford, Calif., 1964.
7. T. IBARAKI, "Complementary Programming," Working Paper, Dept. of Applied Mathematics and Physics, Kyoto University, Kyoto, Japan, 1969.
8. E. L. LAWLER AND D. E. WOOD, "Branch-and-Bound Methods: A Survey," *Opns. Res.* **14**, 699–719 (1966).
9. C. E. LEMKE AND J. T. HOWSON, "Equilibrium Points of Bi-Matrix Games," *SIAM Journal* **12**, 413–423 (1964).
10. S. MUROGA AND T. IBARAKI, "Logical Design of an Optimum Network by Integer Linear Programming I," Rept. No. 264, Dept. of Computer Science, Univ. of Illinois, Urbana, Illinois, 1968.
11. C. C. PETERSEN, "Computational Experience with Variants of the Balas Algorithm Applied to the Selection of *R* and *D* Projects," *Management Sci.* **13**, 736–750 (1967).

Published in Operations Research on October 01, 1971 as DOI: 10.1287/opre.19.6.1523.
This article has not been copyedited or formatted. The final version may differ from this version.