JOURNAL OF RESEARCH of the National Bureau of Standards—B. Mathematics and Mathematical Physics
Vol. 71B, No. 4, October–December 1967

A Pseudo Primal-Dual Integer Programming Algorithm*

Fred Glover**

(November 16, 1966)

The Pseudo Primal-Dual Algorithm solves the pure integer programming problem in two stages, systematically violating and restoring dual feasibility while maintaining an all-integer matrix. The algorithm is related to Gomory All-Integer Algorithm and the Young Primal Integer Programming Algorithm, differing from the former in the dual feasible stage by the choice of cuts and pivot variable, and from the latter in the dual infeasible stage by the use of a more rigid (and faster) rule for restoring dual feasibility.

The net advance in the objective function value produced by the algorithm between two consecutive stages of dual infeasibility is shown to be at least as great as that produced by pivoting with the dual simplex method. Example problems are given that illustrate basic features and variations of the method.

Key Words: Gomory algorithm, integer programming, linear inequalities, maximization.

1. Introduction

The algorithm of this paper alternates between a dual feasible stage related to the Gomory All-Integer Integer Programming Algorithm [4][1] and a dual infeasible stage related to the Young Primal Integer Programming [5]. The Pseudo Primal-Dual algorithm departs from the Gomory and Young algorithms, however, in its choice of cuts and pivot rules, and produces an objective function change between two consecutive stages of dual feasibility at least as great as produced by a pivot with the dual simplex method. In addition, the number of iterations of the dual infeasible stage is less than a particular coefficient in the preceding dual feasible matrix.

Key features and variations of the algorithm are illustrated by detailed solution of example problems in the concluding section.

2. Description of the Problem

Using matrix notation, the problem we are concerned with may be written

$$\text{Maximize } x_0$$
$$\text{Subject to } X = AT,$$

$$X = \begin{bmatrix} x_0 \\ x_1 \\ . \\ . \\ . \\ x_m \end{bmatrix}, \quad T = \begin{bmatrix} 1 \\ -t_1 \\ . \\ . \\ . \\ -t_n \end{bmatrix}, \quad x_i \geqq 0 \text{ for } i = 1, \ldots, m,$$

$$A = (A_0, A, \ldots, A_n) = (a_{ij})$$

$$i = 0, 1, \ldots, m, j = 0, 1, \ldots, n,$$

where the general row equation of $X = AT$ is represented

$$x_i = a_{i0} + \sum_{j=1}^{n} a_{ij}(-t_j), i = 0, 1, \ldots, m \qquad (1)$$

and the last n equations initially have the form

$$x_{m-n+j} = -(-t_j), j = 1, \ldots, n.$$

The above problem represents the ordinary linear programming problem P1 when the x_i may assume fractional values and the pure integer programming problem P2 when the x_i are required to be integers. As is well known, $X = A_0$ provides an optimal solution to P1 when A is both primal and dual feasible, i.e., when $a_{i0} \geqq 0$ for $i = 1, \ldots, m$ and $a_{0j} \geqq 0$ for $j = 1, \ldots, n$. If in addition A_0 is all-integer $X = A_0$ provides an optimal solution to P2.

3. The Dual Simplex Algorithm and the Gomory All-Integer Algorithm

The Dual Simplex Algorithm for solving P1 and the Gomory All-Integer Algorithm for solving P2 are closely related. A basic idea of these methods is to employ a nonsingular transformation of A and T to obtain a new representation $X = \bar{A}\bar{T}$ for X. Thereupon, $\bar{A}$ and $\bar{T}$ assume the role of A and T, and the process repeats until an A matrix is obtained that satisfies the appropriate optimality criteria.

*An invited paper. This report was prepared as part of the activities of the Management Sciences Research Group, Carnegie Institute of Technology, under Contract NONR 760(24) NR 047-048 with the U.S. Office of Naval Research. Reproduction in whole or in part is permitted for any purpose of the U.S. Government.

**Present address: School of Business, University of Texas, Austin, Texas 78712.

[1] Figures in brackets indicate the literature references at the end of this paper.

In applying the Dual Simplex Algorithm, A begins and remains dual feasible. The precise rules of this method are as follows.

The Dual Simplex Algorithm (DSA) for Solving P1

1. If $a_{i0} \geqq 0$ for $i=1, \ldots, m$, then $X=A_0$ is optimal. Otherwise select $r \geqq 1$ such that $a_{r0} < 0$.

2. If $a_{rj} \geqq 0$ for $j=1, \ldots, n$, then P1 has no feasible solution. Otherwise, select $u \geqq 1$ such that $a_{ru} < 0$ and[2] $A_u/a_{ru} >^{l} A_j/a_{rj}$ for all $j \geqq 1$, $j \neq u$, such that $a_{rj} < 0$.

3. Determine $\overline{A}$ by the rules:

$$\overline{A}_u = -A_u/a_{ru}$$

$$\overline{A}_j = A_j - A_u(a_{rj}/a_{ru}) \text{ if } j \neq u.$$

4. Let $\bar{t}_u \equiv x_r$ and $\bar{t}_j \equiv t_j$ for $j \neq u$. Designate $\overline{A}$ and $\overline{T}$ to be the current A matrix and T vector and return to 1.

The All-Integer Integer Programming Algorithm of Ralph Gomory modifies the DSA by introducing new equations called cuts to give the transformation of A into $\overline{A}$. Specifically, in the context of P2, eq (1) implies the cut[3]

$$s_i = [a_{i0}/\lambda] + \sum_{j=1}^{n} [a_{ij}/\lambda](-t_j) \qquad (2)$$

where s_i is a nonegative integer variable.

For an appropriate value of $\lambda > 0$, one may use (2) to determine A by replacing each occurrence of a_{rj} and a_{ru} in instruction 2 and 3 of the DSA by $[a_{rj}/\lambda]$ and $[a_{ru}/\lambda]$. The initial A matrix is assumed not only to be lexicographically dual feasible, but also to consist entirely of integer coefficients. To keep $\overline{A}$ all-integer it evidently suffices to select λ so that $[a_{ru}/\lambda] = -1$ (although the index u may not be the same for the DSA and the all-integer algorithm). Thus, the Gomory algorithm can be described as follows.

The Gomory All-Integer Algorithm

1. If $a_{i0} \geqq 0$ for $i=1, \ldots, m$, then $X=A_0$ is optimal. Otherwise, select $r \geqq 1$ so that $a_{r0} < 0$.

2. If $a_{rj} \geqq 0$ for $j=1, \ldots, n$, then P2 has no feasible solution. Otherwise, define $a'_{rj} = [a_{rj}/\lambda]$, $j=0, \ldots, n$. Select $v \geqq 1$ and $\lambda > 0$ so that $a'_{rv} = -1$ and $-A_v >^{l} A_j/a'_{rj}$ for all $j \geqq 1$, $j \neq v$, such that $a'_{rj} \leqq -1$.

3. Determine $\overline{A}$ so that

$$\overline{A}_v = A_v;\ \overline{A}_j = A_j + a'_{rj}\ A_v\ j=0, 1, \ldots, n, j \neq v.$$

4. Let $\bar{t}_v \equiv s_r$ and[4] $\bar{t}_j \equiv t_j$ for $j \neq v$. Designate $\overline{A}$ and $\overline{T}$ to be the current A and T and return to 1.

It is evident that the Gomory algorithm results simply by applying instructions 2, 3, and 4 of the DSA to the cut eq (2) instead of the source eq (1) (for $i=r$). Simplifications in the computation of λ and v are given by Gomory in [4], where it is shown that finite convergence is guaranteed by periodically selecting r in instruction 1 to be the least $i \geqq 1$ such that $a_{i0} < 0$.

The fundamental ideas underlying the all-integer algorithm and the DSA provide the conceptual starting points for the Pseudo Primal-Dual Algorithm, whose strategy and special characteristics we develop to follow.

4. The Pseudo Primal-Dual Algorithm

The Pseudo Primal Dual Algorithm involves a sequence of "major iterations," each of which consists of several pivot steps using cuts of the form (2) derived from a single source row (or value of i). Each major iteration is divided into 2 stages. The first stage consists of a single pivot step. However, instead of selecting the pivot column v by applying the DSA criterion to (2) (as in the all-integer algorithm), the method selects v to be the same as u, i.e., by applying the DSA criterion to (1). In addition, λ no longer truly serves as a parameter, but is always $-a_{ru}$.[5] If Stage 1 does not destroy dual feasibility, then Stage 2 is vacuous. Otherwise dual feasibility is restored by a sequence of "pseudo-primal" pivot steps using the column that is lexicographically most negative when divided by the corresponding coefficient in the source row (restricting attention to positive coefficients).

To specify the algorithm more precisely, we introduce the following additional notation. Relative to a selected equation r, let

$$A_j^* = A_j/a_{rj},\ j=1, \ldots, n, \text{ provided } a_{rj} \neq 0.$$

(Likewise, for the matrix $\overline{A}$ we define $\overline{A}_j^* = \overline{A}_j/\overline{a}_{rj}$.)

Let u be determined so that[6]

$$u \geqq 1,\ a_{ru} < 0 \text{ and } A_u^* >^{l} A_j^* \text{ for all } j \geqq 1, j \neq u \text{ such that } a_{rj} < 0. \qquad (3)$$

[2] A vector α is defined to be lexicographically larger than a vector β (symbolized $\alpha >^{l} \beta$ or $\beta <^{l} \alpha$) if the first nonzero component of α-β is positive. The condition of instruction 2 implies the more familiar condition $a_{0u}/a_{ru} = \operatorname{Max}_{j \geqq 1} \{a_{0j}/a_{rj} : a_{rj} < 0\}$, and also provides a rule that assures a finite algorithm in the case of degeneracy. It is assumed here that A begins lexicographically dual feasible; that is, $A_j >^{l} 0$ for all $j \geqq 1$. Note that $A_p/a_{rp} = A_q/a_{rq}$ for $p \neq q$ and $p, q \geqq 1$ is impossible since the initial A_j vectors for $j \geqq 1$ contain the $-I$ matrix, hence these A_j begin and remain linearly independent.

[3] $[y]$ denotes the greatest integer $\leqq y$.

[4] The cut variable s_r should in strictness be additionally subscripted (e.g., with the iteration number) to avoid ambiguity; s_3 from iteration 5 may not be the same variable as s_3 from iteration 9.

[5] Motivation for these choices is provided in [2], where it is shown that once u is selected, selecting $\lambda \neq -a_{ru}$ may be interpreted as applying the Bound Escalation Method [1] to an equation that is less constraining than (1). One of the consequences of this is manifested in Theorem 5, below.

[6] Note that this definition corresponds to the one given for u in instruction 2 of the DSA.

In addition, let s be determined so that

$$a_{rs} > 0,\ s \geqq 1, \text{ and } A_s^* \overset{l}{<} A_j^* \text{ for all } j \geqq 1,\ j \neq s,$$
$$\text{such that } a_{rj} > 0. \tag{4}$$

Beginning with A all integer and lexicographically dual feasible, the Pseudo Primal-Dual Algorithm is then as follows.

The Pseudo Primal-Dual Algorithm (PPDA)

STAGE 1

1. If $a_{i0} \geqq 0$ for $i=1, \ldots, m$, then $X=A_0$ is optimal. Otherwise, select $r \geqq 1$ such that $a_{r0} < 0$ (periodically, $r = \text{Min } \{i : a_{i0} < 0\}$).

2. If $a_{rj} \geqq 0$ for all $j \geqq 1$, then P2 has no feasible solution. Otherwise, identify u by (3).

3. Determine $\bar{A}$ so that

$$\bar{A}_u = A_u$$

$$\bar{A}_j = A_j + [a_{rj}/(-a_{ru})]\, A_u \qquad j=0, \ldots, n \qquad j \neq u.$$

4. Let $\bar{t}_u \equiv s_r$ and $\bar{t}_j \equiv t_j$ for $j \neq u$. Designate $\bar{A}$ and $\bar{T}$ to be the current A matrix and T vector. If $A_j \overset{l}{>} 0$ for all $j \geqq 1$, return to instruction 1. Otherwise,

STAGE 2

5. Retain the index r unchanged, identify s by (4), and let

$$\bar{A}_s = -A_s$$
$$\bar{A}_j = A_j - [a_{rj}/a_{rs}]\, A_s \qquad j=0, \ldots, n \qquad j \neq s.$$

6. Let $\bar{t}_s \equiv s_r$ and $\bar{t}_j \equiv t_j$ for $j \neq s$. Designate $\bar{A}$ and $\bar{T}$ to be the current A and T. If $A_j \overset{l}{>} 0$ for all $j \geqq 1$, return to instruction 1. Otherwise, return to instruction 5.

We observe that instruction 3 of the PPDA employs the Gomory cut (2) for $i=r$ and $\lambda = -a_{ru}$, while instruction 5 employs (2) for $i=r$ and $\lambda = a_{rs}$. Also the definition of s assures $A_s \overset{l}{<} 0$, if there exists a $j \geqq 1$ such that $A_j \overset{l}{<} 0$ and $a_{rj} > 0$.

THEOREMS AND PROOFS [7]

To justify the algorithm and develop its properties, we will undertake to establish the validity of two very simple and important relationships for every A matrix it generates:

$$A_u^* \overset{l}{<} A_s^* \text{ [8]} \tag{5}$$

$$(a_{rj} = 0) \Rightarrow A_j \overset{l}{>} 0. \tag{6}$$

We observe to start that (5) and (6) must hold whenever instruction 3 is initiated since the fact that A is always lexicographically dual feasible in Stage 1 implies $A_j \overset{l}{>} 0$ for all $j \geqq 1$, hence $A_u^* \overset{l}{<} 0$ and $A_s^* \overset{l}{>} 0$.

To show that (5) and (6) hold throughout the algorithm we introduce

LEMMA 1. *Let* $\bar{A}_w = K_w A_w$ *and* $\bar{A}_j = A_j - K_j A_w$ $(j \neq w)$, *for any scalars* K_j, K_w *such that* $K_w \neq 0$. *Then, if* $a_{rw} \neq 0$

$$a_{rj} A_w^* \overset{l}{<} A_j \tag{7}$$

if and only if

$$\bar{a}_{rj} \bar{A}_w^* \overset{l}{<} \bar{A}_j. \tag{8}$$

From the definitions,

PROOF: $\bar{a}_{rj} A_w^* = \bar{a}_{rj} A_w^* = (a_{rj} - K_j a_{rw}) A_w^* = a_{rj} A_w^* - K_j A_w$. Also $\bar{A}_j = A_j - K_j A_w$. Thus $\bar{a}_{rj} A_w^* - \bar{A}_j = a_{rj} A_w^* - A_j$, and the lemma follows at once.[9]

The definitions of $\bar{A}_w$ and $\bar{A}_j$ in Lemma 1 may be seen to accord with the definitions of $\bar{A}_w$ and $\bar{A}_j$ at instruction 3 and 5 of the algorithm for $w=u$ and $w=s$ respectively. Furthermore, if $a_{rj} < 0$, then condition (7) of Lemma 1 is the same as $A_w^* \overset{l}{>} A_j^*$, permitting w to be identified with u, while if $a_{rj} > 0$, then (7) is the same as $A_w^* \overset{l}{<} A_j^*$, permitting w to be identified with s. With these observations as a foundation, we state and prove the key result alluded to above.

THEOREM 1: *Let* $\bar{A}_w$ *and* $\bar{A}_j$ *be given as in Lemma 1 for* $w=r$ *or* $w=s$, *and let* $\bar{u}$ *and* $\bar{s}$ *be defined relative to the matrix* $\bar{A}$ *in the same way that* u *and* s *are defined relative to* A. *Then, if*

(5) $A_u^* \overset{l}{<} A_s^*$, *and* (6) $(a_{rj}=0) \Rightarrow A_j \overset{l}{>} 0$ *for all* $j \geqq 1$

it follows that

$(\bar{5})$ $\bar{A}_u^* \overset{l}{<} \bar{A}_s^*$, *and* $(\bar{6})$ $(a_{rj}=0) \Rightarrow \bar{A}_j \overset{l}{>} 0$ *for all* $j \geqq 1$.

PROOF: Letting either $w=u$ or $w=s$, $A_u^* \overset{l}{<} A_s^*$ is equivalent to

$$A_p^* \overset{l}{<} A_w^* \overset{l}{<} A_q^* \text{ for all } p, q \geqq 1,\ p, q \neq w,$$
$$\text{such that } a_{rp} < 0 \text{ and } a_{rq} > 0.$$

This immediately implies (7) of Lemma 1 for $a_{rj} \neq 0$, and (6) implies (7) for $a_{rj}=0$. Thus (8) of Lemma 1 is true, which establishes (6) and

$$\bar{A}_p^* \overset{l}{<} \bar{A}_w^* \overset{l}{<} \bar{A}^* \text{ for all } p, q \geqq 1$$
$$p, q \neq w, \text{ such that } \bar{a}_{rp} < 0 \text{ and } \bar{a}_{rq} > 0.$$

But the existence of a w satisfying this last relation-

[7] These wishing to defer consideration of the theorems and proofs can skip to the preliminary illustrative material of the next section with few sacrifices in understanding.

[8] By convention, we assume that $A^* \overset{l}{<} A_s^*$ holds trivially if either u or s is not well-defined.

[9] This proof evidently permits the lexicographic inequality signs in (6) and (7) to be reversed or replaced by equality signs. Also, the requirement $a_{rw} \neq 0$ can be dropped after multiplying (6) through by a_{rw} and (7) through by a_{rw}, thus replacing A_w^* and $\bar{A}_w^*$ with A_w and $\bar{A}_w$.

ship also implies that it must hold for $w=\bar{u}$ and $w=\bar{s}$, and hence is equivalent to (5).

We now restrict $\overline{A}_w$ and $\overline{A}_j$ given in Lemma 1 to bring them into closer correspondence with the definitions provided by the algorithm. In doing so we establish the additional results required to demonstrate the properties claimed for the method in section 1.

COROLLARY 1: *Let* $\overline{\mathrm{A}}_{\mathrm{w}}$ *and* $\overline{\mathrm{A}}_{\mathrm{j}}$ *be given as in Lemma 1. If* $\mathrm{K_w}<0$ *for* $\mathrm{w=s}$, *and* $\mathrm{K_w}>0$ *for* $\mathrm{w=u}$, *then* (5) *and* (6) *imply*

$$A_w^* \overset{l}{<} \overline{A}_s^*.$$

PROOF: Since $a_{rs}>0$ and $a_{ru}<0$, the sign restrictions on K_w for $w=u$ and $w=s$ imply $\overline{a}_{rw}<0$, and hence either $\overline{A}_w^* \overset{l}{<} \overline{A}_{\bar{u}}^*$ or $w=\bar{u}$. The relations $A_{\bar{w}}^*=\overline{A}_w^*$ and $\overline{A}_{\bar{u}}^* \overset{l}{<} \overline{A}_{\bar{s}}^*$ immediately establish the corollary.

COROLLARY 2: *Let* $\overline{\mathrm{A}}_{\mathrm{w}}$ *and* $\overline{\mathrm{A}}_{\mathrm{j}}$ *be given as in Lemma 1. If* $\mathrm{A}_{\mathrm{w}}^* \overset{l}{<} 0$ *for* $\mathrm{w=u}$ *or* $\mathrm{w=s}$, *then* (5) *and* (6) *imply*

$$\mathrm{A_j} \overset{l}{<} 0 \Rightarrow \mathrm{a_{rj}}>0, \quad \overline{\mathrm{A}}_{\mathrm{j}} \overset{l}{<} 0 \Rightarrow \overline{\mathrm{a}}_{\mathrm{rj}}>0 \text{ for all } \mathrm{j} \geqq 1, \tag{9}$$

and

$$\mathrm{A}_{\mathrm{u}}^*, \mathrm{A}_{\mathrm{s}}^*, \overline{\mathrm{A}}_{\bar{\mathrm{u}}}^*, \text{and } \mathrm{A}_{\bar{\mathrm{s}}}^* \tag{10}$$

are lexicographically negative (*if they exist*).

PROOF: By the proof of Theorem 1, (5) and (6) imply (7) of Lemma 1 for both $w=u$ and $w=s$. Thus if $A_w^* \overset{l}{<} 0$ for either $w=u$ or $w=s$, it follows that $A_w \overset{l}{<} 0 \Rightarrow a_{rw}>0$, since $A_u^* \overset{l}{<} 0 \Rightarrow A_u \overset{l}{>} 0$, and $a_{rs}>0$ holds by the definition of s. Also, $A_j \overset{l}{<} 0 \Rightarrow a_{rj}>0$ for $j \neq w$ by (7) of Theorem 1. Then, since (7) implies (8), and $A_{\bar{w}}^*=\overline{A}_w^*$, the same argument applied to (8) yields the second half of (9) above. Finally (10) follows from (9).

Corollary 1 establishes the important fact that A_s^* (letting $w=s$) is always lexicographically increasing in Stage 2 of the PPDA. Corollary 2 implies that s will always be meaningfully defined at instruction 5 of the algorithm. We require one additional result to establish finiteness for Stage 2.

THEOREM 2: [10] *For* $\overline{\mathrm{A}}$ *given by instruction 3 of the PPDA*,

$$-\mathrm{a_{ru}} > \overline{\mathrm{a}}_{\mathrm{rj}} \geqq 0 \text{ for all } \mathrm{j} \neq \mathrm{u}$$

and for $\overline{\mathrm{A}}$ *given by instruction 5 of the PPDA*,

$$\mathrm{a_{rs}} > \overline{\mathrm{a}}_{\mathrm{rj}} \geqq 0 \text{ for all } \mathrm{j} \neq \mathrm{s}.$$

PROOF: From instruction 5, $\overline{a}_{rj}=a_{rj}-[a_{rj}/a_{rs}]a_{rs}$. Since $y \geqq [y] > y-1$ for all numbers y, and

$$a_{rs}>0,\ a_{rj}-(a_{rj}-a_{rs}) > a_{rj} \geqq a_{rj}-a_{rj}.$$

The proof for $\overline{A}$ given by instruction 3 is analogous.

THEOREM 3. *Let* δ *be the value of* $\mathrm{a_{rs}}$ *on the first visit in instruction 5 during any execution of Stage 2. Then instruction 5 will be visited at most* δ *times before the current execution of Stage 2 is terminated with* $\mathrm{A_j} \overset{l}{>} 0$ *for all* $\mathrm{j} \geqq 1$.

PROOF: From instruction 5, $\overline{a}_{rs}=-a_{rs}$. Hence by Theorem 2, $\overline{a}_{rs}<a_{rs}$. This can occur at most δ times, since if s is still meaningfully defined at the $\delta+1$st step, then $\overline{a}_{r\bar{s}}=0$, which is impossible. But by Corollary 2 to Theorem 2, s must be meaningfully defined unless $A_j \overset{l}{>} 0$ for all $j \geqq 1$. This completes the proof.[11]

It should be noted that, while the value of δ given by the preceding theorem provides an upper bound on the number of iterations of Stage 2, a larger value of δ will not necessarily entail a greater number of iterations than a smaller one.

Before stating and proving the theorem that establishes finiteness for the complete algorithm, we give two theorems that disclose additional properties of Stages 1 and 2.

THEOREM 4: *Let* $\mathrm{w=u}$ *if* $\overline{\mathrm{A}}$ *is defined by instruction 3 of the PPDA and* $\mathrm{w=s}$ *if* $\overline{\mathrm{A}}$ *is defined by instruction 5. Then*

$$\overline{\mathrm{A}}_{\mathrm{j}} \overset{l}{<} 0,\ \mathrm{j} \neq \mathrm{w}$$

implies

$$-\overline{\mathrm{A}}_{\mathrm{w}} \overset{l}{<} \overline{\mathrm{A}}_{\mathrm{j}} \text{ and } -\overline{\mathrm{a}}_{\mathrm{pw}} < \overline{\mathrm{a}}_{\mathrm{pj}},$$

where

$$\mathrm{p} = Min\ (\mathrm{i}: \overline{\mathrm{a}}_{\mathrm{iw}} \neq 0).$$

PROOF: By Corollary 2 to Theorem 1 $\overline{a}_{rj}<0$, and hence by the definition of $\bar{s}$, $\overline{A}_{\bar{s}}^* \overset{l}{<} \overline{A}_j^*$ or $j=\bar{s}$. From Corollary 1 to Theorem 1 it follows that $A_w^*<\overline{A}_j^*$, and $a_{0w}/a_{rw} \leqq \overline{a}_{0j}/\overline{a}_{rj}$. If $\mathrm{a_{0w}}=0$, then $\overline{a}_{rj}=0$ is implied by $\overline{A}_j<0$, and from $\overline{A}_w=K_wA_w$ we conclude $\mathrm{a_{pw}}/a_{rw} \leqq \overline{a}_{pj}/\overline{a}_{rj}$ and

$$\overline{a}_{ij}=a_{iw}=\overline{a}_{iw}=0 \text{ for } i<p. \text{ If } \overline{a}_{pj}=0,$$

the theorem is immediately true. Thus, suppose $\overline{a}_{pj}<0$. For $w=s$, we have $a_{rs}>\overline{a}_{rj}$ (Theorem 2) and $a_{ps}=-\overline{a}_{ps}<0$. Consequently $-\overline{a}_{ps}<\overline{a}_{pj}$. The Theorem is similarly proved for $w=u$.

Theorem 4 implies that dual feasibility (though not necessarily lexicographic dual feasibility) must be restored in Stage 2 in at most $-\mathrm{a_{0s}}$ iterations, since $a_{0s}<\overline{a}_{0\bar{s}}$ must occur at every visit to instruction 5 as long as $a_{0s}<0$. This rate of progression toward dual feasibility in Stage 2 is significant in that it exceeds any that can be proved for the primal all-integer algorithms.[12]

[10] An equivalent result was first given in the context of a primal algorithm by R. D. Young [5] and in the context of a dual algorithm at about the same time by the author in [1]. An interesting and easily proved consequence of this theorem, which we do not exploit here, is that the "converse" of Theorem 2 is valid when $\overline{A}$ is given by instruction 5 of the algorithm with w replacing s at that instruction; i.e., if $\overline{A}_j=A_j-[a_{rj}/a_{rw}]A_w$, $\overline{A}_w=-A_w$, then (5) and (6) imply (5), (6) and $w=s$. A corresponding result holds when $\overline{A}$ is given by instruction 3 with w replacing u.

[11] More restrictively δ can be the value of a_{rs} divided by the greatest common divisor of the a_{rj} for $j \geqq 1$. Also, note that Theorem 2 implies $-a_{ru}>\delta$, where a_{ru} is given at instruction 3, thus providing a known upper bound on the number of iterations in Stage 2 before it is initiated.

[12] The form of this progression has also led to a choice rule that guarantees finite convergence for a simplified primal method [3].

Our next Theorem shows that the advance (decrease) in the objective function value in Stage 1 is always greater than or equal to that produced by the Gomory all-integer algorithm.

THEOREM 5: *Let* $\bar{a}^1_{00}$ *be the value of* $\bar{a}_{00}$ *obtained at instruction 3 of the Gomory algorithm and* $\bar{a}^2_{00}$ *be the value of* $\bar{a}_{00}$ *obtained at instruction 3 of the PPDA (for the same choice of* r*). Then* $\bar{a}^2_{00} \leqq \bar{a}^1_{00}$.

PROOF: $\bar{a}^1_{00} = a_{00} + a_{0v}[a_{r0}/\lambda]$ and

$$\bar{a}^2_{00} = a_{00} + a_{0u}[a_{r0}/(-a_{ru})].$$

From the choice of λ and v specified in instruction 2 of the Gomory algorithm, we have

$$-[a_{ru}/\lambda] \leqq a_{0u}/a_{0v},$$

hence

$$-[a_{ru}/\lambda] \leqq [a_{0u}/a_{0v}], \text{ and } a_{ru}/\lambda \geqq -[a_{0u}/a_{0v}].$$

Since $a_{r0} < 0$ and

$$a_{ru} < 0,\ a_{r0}/\lambda \geqq -(a_{r0}/r_u)\ [a_{0u}/a_{0v}] \geqq [a_{r0}/(-a_{ru})][a^{0u}/a_{0v}].$$

But $a_{0v}[a_{0v}/a_{0v}] \leqq a_{0u}$ and $[a_{r0}/(-a_{ru})] < 0$, and thus $a_{0v}[a_{r0}/\lambda] \geqq a_{0u}[a_{r0}/(-a_{ru})]$. This implies $\bar{a}^1_{00} \geqq \bar{a}^2_{00}$, completing the proof.

We now show that the net lexicographic decrease in A_0 brought about by the PPDA is at least as great as that produced by the dual simplex algorithm. As remarked earlier, this immediately implies that the PPDA is finite, thereby completing the justification of the algorithm.

THEOREM 6: *Given the same* A *matrix and the same choice of* r*, the amount of the lexicographic decrease in* A_0 *resulting from two successive visits to instruction 1 of the PPDA equals or exceeds that resulting from two successive visits to instruction 1 of the DSA.*

PROOF: Let A^0 denote the A matrix on the first of two consecutive visits to instruction 1 of the PPDA, and let $\overline{\overline{A}}$ denote A on the second of these visits. In addition, for each iteration h at instruction 5, let $A^h (h \geqq 1)$ denote the current A matrix and let $K^h = [a^h_{r0}/a^h_{rs}]$.[13] Then we may write

$$\overline{\overline{A}}_0 = A^1_0 - \sum_{h \geqq 2} K^h A^h_s \tag{11}$$

$$\overline{\overline{a}}_{r0} = a^1_{r0} - \sum_{h \geqq 2} K^h a^h_{rs}. \tag{12}$$

By the definition of u, the A_0 vector that replaces A^0_0 on the second of the two visits to instruction 1 of the DSA is $A^0_0 - a^0_{r0}A^{0*}_u$. (We note that this results in a lexicographic decrease in A_0 since $a^0_{ro} < 0$ and $A^{0*}_u < 0$.) Thus, to prove the Theorem we must show that $\overline{\overline{A}}_0 \overset{l}{<} A^0_0 - a^0_{r0}A^{0*}_u$.

Since $\overline{\overline{a}}_{r0} \geqq 0$, from (12) we obtain

$$a^1_{r0} \geqq \sum_{h \geqq 2} K^h a^h_{rs}. \tag{13}$$

Also, from the definition of A^{h*}_s,

$$\sum_{h \geqq 2} K^h A^h_s = \sum_{h \geqq 2} K^h a^h_{rs} A^{h*}_s. \tag{14}$$

By Corollary 1 to Theorem 1, it follows that $A^{h*}_s \overset{l}{>} A^{0*}_u$ for all $h \geqq 1$. Since $K^h \geqq 0$ and $a^h_{rs} \geqq 1$, (14) implies

$$\sum_{h \geqq 2} K^h A^h_s \overset{l}{\geqq} A^{0*}_u \sum_{h \geqq 2} K^h a^h_{rs}.$$

Also, since $A^{0*}_u \overset{l}{<} 0$, we have by (13) that

$$a^1_{r0}A^{0*}_u \leqq A^{0*}_u \sum_{h \geqq 2} a^h_{rs}.$$

Thus, from (11) we conclude

$$\overline{\overline{A}}_0 \overset{l}{\leqq} A^1_0 - a^1_{r0}A^{0*}_u. \tag{15}$$

Finally, using the fact that $\bar{a}_{ru}A^{0*}_u = \bar{A}^0_u$, the definitions of $\bar{A}^0_0$ and $\bar{a}^0_{r0}$ yield

$$\bar{A}^0_0 - \bar{a}^0_{r0}A^{0*}_u = A^0_0 - a^0_{r0}A^{0*}_u. \tag{16}$$

But $A^1 = \bar{A}^0$, hence (15) and (16) establish the desired result.

5. Example Problems and Comments

Three problems are solved in this section to illustrate the fundamental characteristics of the PPDA. In addition, variations of the PPDA are developed by informal example.

EXAMPLE PROBLEM 1: [14]

$$\text{Maximize } x_0 = 0 + 23(-x_3) + 17(-x_4) + 3(-x_5) + 7(-x_6)$$

$$\text{subj. to } x_1 = -128 - 27(-x_3) - 20(-x_4) - 16(-x_5) - 17(-x_6)$$

$$x_2 = -45 - 22(-x_3) - 14(-x_4) + 9(-x_5) + 2(-x_6),\ x_j \geqq 0 \text{ for } j \geqq 1.$$

[13] We do not bother to represent the fact that s depends on h.

[14] This problem may also be written in the form

Minimize $23x_3 + 17x_4 + 3x_5 + 7x_6$
s.t. $27x_3 + 20x_4 + 16x_5 + 17x_6 \geqq 128$
$22x_3 + 14x_4 - 9x_5 - 2x_6 \geqq 45$

for nonnegative integer x_j, where x_1 and x_2 are introduced as slack variables to change the inequalities into equalities.

Representing $X=AT$ in detached coefficient (i.e., tableau) form, we have

0.

	1	$-x_3$ ↓	$-x_4$	$-x_5$	$-x_6$
$x_0=$	0	23	17	3	7
$x_1=$	−128	−27	−20	−16	−17
→ $x_2=$	−45	−22	−14	9	2
$x_3=$	0	−1	0	0	0
$x_4=$	0	0	−1	0	0
$x_5=$	0	0	0	−1	0
$x_6=$	0	0	0	0	−1

Beginning with instruction 1 of the PPDA, we observe that $a_{i0}<0$ for $i=1$ and 2, and hence proceed to instruction 2. Equation r is selected at this step to be the one with the fewest negative components.[15] Thus, $r=2$, as indicated by the arrow in the tableau above pointing to row 2. From the definition of A_u^*, $u=1$ when $r=2$, hence the arrow pointing to column 1 (A_1). Instructions 3 and 4 then yield the new tableau

1.

	1	$-s_2$	$-x_4$ ↓	$-x_5$	$-x_6$
$x_0=$	−69	23	−6	3	7
$x_1=$	−47	−27	7	−16	−17
→ $x_2=$	21	−22	8	9	2
$x_3=$	3	−1	1	0	0
$x_4=$	0	0	−1	0	0
$x_5=$	0	0	0	−1	0
$x_6=$	0	0	0	0	1

In this tableau $A_2 \overset{l}{<} 0$. Therefore, with r still at 2, we proceed to instruction 5 and apply the indicated transformation for $s=2$. The updated tableau obtained at instruction 6 is then

2.

				↓	
	−57	5	6	9	7
→	−61	−6	−7	−23	−17
	5	2	−8	1	2
	1	2	−1	−1	0
	2	−3	1	1	0
	0	0	0	−1	0
	0	0	0	0	−1

Since the x_i along the left margin are unchanging and the t_j along the top margin are irrelevant, we do not bother in this and subsequent tableaus to specify the identity of these variables. In Tableau 2, $A_j \overset{l}{>} 0$ for all $j \geqq 1$, requiring a return to instruction 1. Since $a_{i0}=-61<0$, instruction 2 is visited next, and $r=1$ is the only choice. Now $u=3$, and by instructions 3 and 4 we obtain

3.

					↓
	−84	−4	−3	9	−2
→	8	17	16	−23	6
	2	1	−9	1	1
	4	3	0	−1	1
	−1	−4	0	1	−1
	3	1	1	−1	1
	0	0	0	0	−1

Once again A is dual infeasible. From its definition, $s=4$, thereby at instructions 5 and 6 yielding the tableau

4.

−82	0	1	1	2
2	5	4	1	−6
1	−1	−11	5	−1
3	1	−2	3	−1
0	−2	2	−3	1
2	−1	−1	3	−1
1	2	2	−4	1

A is now both primal and dual feasible, and the problem is solved. From $X=A_0$ we obtain the optimal solution $x_0=-82$, $x_1=2$, $x_2=1$, $x_3=3$, $x_4=0$, $x_5=2$, $x_6=1$.

The next example problem illustrates additional features of the algorithm.

EXAMPLE PROBLEM 2:

Maximize $x_0=0+3(-x_2)+5(-x_3)$

$$+9(-x_4)+7(-x_5)+13(-x_6)$$

subj. to $x_1=-398-6(-x_2)-15(-x_3)-36(-x_4)$

$$-23(-x_5)-41(-x_6)\ x_j \geqq 0\ j=1, \ldots, 6.$$

For convenience we will not bother to write down the last five rows of the tableau corresponding to the $-I$ matrix and zero vector, but will explicitly represent these rows only when they are changed from their original form. Thus, for the initial tableau we have

0.

				↓		
$x_0=$	0	3	5	9	7	13
→ $x_1=$	−398	−6	−15	−36	−23	−41

The arrows accompanying the tableau point to row r and column u. Since $u=3$, the third row of the original $-I$ matrix (corresponding to x_4) will be modified

[15] Our choice here is based on considerations developed in [1].

by the transformation defined at instruction 3. Thus, in the resulting tableau below this modified row is included following the modified rows 0 and 1. We keep track of the components of X in the left margin since their order has been shuffled by our bookkeeping conventions.

1.

		↓				
$x_0=$	−108	−6	−4	9	−2	−5
$x_1=$	34	30	21	−36	13	31
$x_4=$	12	1	1	−1	1	2

At instruction 5 the transformations are initiated to restore A to dual feasibility. Since $s=1$, the first row of the original $-I$ matrix (corresponding to x_2) will now be changed, yielding the new last row in the tableau below.

2.

		↓				
$x_0=$	−102	6	−4	−3	−2	1
$\rightarrow x_1=$	4	−30	21	24	13	1
$x_4=$	11	−1	1	1	1	1
$x_2=$	1	1	0	−2	0	1

This tableau is still dual infeasible and instruction 5 must therefore be repeated.

3.

		↓				
$x_0=$	−102	−2	4	1	−2	−1
$\rightarrow x_1=$	4	12	−21	3	13	1
$x_4=$	11	1	−1	0	1	1
$x_2=$	1	1	0	−2	0	1
$x_3=$	0	−2	1	1	0	0

Once again repeating instruction 5 we obtain

4.

$x_0=$	−102	2	0	1	0	1
$x_1=$	4	−12	3	3	1	1
$x_4=$	11	−1	1	0	0	1
$x_2=$	1	−1	2	−2	−1	1
$x_3=$	0	2	−3	1	2	0

The problem is now solved, and an optimal solution is given by $x_0=-102$, $x_1=4$, $x_4=11$, $x_2=1$, and $x_3=x_5=x_6=0$.

For the proceding problem, we note that the optimal solution was already given in Tableau 2. Since A was not dual feasible, however, the solution was not identified as optimal at that point. Nevertheless, it would have been possible to make this identification in the following way.

We create a new column from A_s $(s=2)$ in Tableau 2 by dividing A_s through by $-a_{0s}$ $(=4)$ [16] and then adjoin this column to the right of the others in the tableau. Since the variable associated with this column (call it z) must be zero in the final solution, we also adjoin the two equations to assure $z \geqq 0$ and $-z \geqq 0$ at the end of the tableau. Thus we obtain

2′.

						↓
$x_0=$	−102	6	−4	−3	−2	1	−1
$\rightarrow x_1=$	4	−30	21	24	13	1	$\frac{21}{4}$
$x_4=$	11	−1	1	1	1	1	$\frac{1}{4}$
$x_2=$	1	1	0	−2	0	1	0
$z=$	0	0	0	0	0	0	−1
$-z=$	0	0	0	0	0	0	1

The new column is segregated by the added partition. It is evident by its construction that this column must qualify as the new A_s.[17]

It is unnecessary to carry out the computations at instruction 5 in order to predict two things about the matrix $\bar{A}$ that will result.

First, since the first component of the new column in Tableau 2′ is -1 and the components of row 0 are integers,[18] it follows from Theorem 4 that $\bar{a}_{0j}=0$ for all $j \geqq 1$ such that $\bar{A}_j \lneq 0$. Consequently, then, $\bar{A}$ must be dual feasible (though perhaps not lexicographically dual feasible).

The second thing to observe is that $[a_{r0}/a_{rs}]=0$ $\left(\text{for } a_{r0}=4 \text{ and } a_{rs}=\frac{21}{4}\right)$, and hence $\bar{A}_0=A_0$. This fact and the one just established assure that the (primal) feasible solution values for the x_i given in Tableau 2 (and 2′) must also be optimal. In short, we have established that $[a_{r0}/a_{rs}]=0$ is a sufficient condition for a feasible solution to be optimal.

If adjoined rows and columns are actually employed in solving the problem, and not simply as a means of checking for optimality, then it will eventually be possible to restore the tableau to its original size.[19] This approach of adjoining columns may also be used at Step 2 to prevent A from becoming dual infeasible in the first place. There are clearly a number of possible variations, and by following appropriate rules the tableau need not be expanded to the extent depicted by our illustration each time a new variable is added. To insure convergence it is of course necessary to forbid the addition of an unlimited number of rows and columns to the tableau.

Our last example problem is a very simple one that poses considerable difficulty for Stage 2 of the PPDA.

[16] If $a_{0s}=0$, we instead divide through by the negative of the first nonzero component of A_s.

[17] More generally, if course, we could adjoin any column $A_h \lneq 0$ (to qualify as the new A_s) such that $A_u^* \lneq A_h^* \lneq A_v^*$.

[18] By permitting rational numbers in the tableau, it suffices more generally to select the first component of the adjoined row to be $-1/k$, where ka_{ij} is an integer for all i and j. If row 0 already consists of negative components, then our remarks have reference instead to the first row i such that $a_{is} \neq 0$. To demonstrate that a feasible solution is optimal, however, consideration may be limited as above to row 0.

[19] By selecting row r to be one or the other of the adjoined rows (which will always be negative of each other), and persisting in this, eventually there will remain only one column of the tableau with nonzero components in these rows, at which time the indicated column and rows may be dropped. The optimal solution may of course be obtained before this size reduction process is completed.

(We will later show how to overcome the difficulty by making the algorithm more flexible than the version of section 3.)

EXAMPLE PROBLEM 3:

Maximize $x_0 = 0 + 1(-x_2) + 28(-x_3)$

subj. to $x_1 = -98 - 1(-x_2) - 45(-x_3)$

$x_j \geqq 0$ for $j \geqq 1$.

0.

0	1	28
−98	−1	−45
0	−1	0
0	0	−1

The next three tableaus are written without additional comment.

1.

−84	−27	28
37	44	−45
0	−1	0
3	1	−1

2.

−84	27	−26
37	−44	43
0	1	−2
3	−1	1

3.

−84	−25	26
37	42	−43
0	−3	2
3	1	−1

One may infer from the structure of this problem that after six more steps we will obtain

−64	−19	20
0	36	−37
8	−9	8
2	1	−1

This tableau gives an optimal solution by the remarks relating to the previous example problem. However, to restore dual feasibility we may project by inference that 20 additional iterations of Stage 2 are required, at which point we obtain

−64	1	0
0	−18	17
8	27	−28
2	−1	1

Two interacting features of Tableau 1 bequeathed by Stage 1 appear to have contributed to the difficulty encountered in Stage 2: (i) A_1^* and A_2^* are nearly the same, and (ii) the components a_{03} and a_{r3} of the vector $A_3 = A_1 + A_2$ are small in absolute value relative to the corresponding components of both A_1 and A_2.[20]

Conditions such as these may be taken to indicate that the transformation employed in Stage 1 should be modified to provide a different A matrix for Stage 2. Specifically, we interpret (i) and (ii) to imply that the choice of u in Tableau 0 would better be given by $u = 1$ instead of $u = 2$.

How is such an altered choice possible? Note that, if the coefficient of a_{11} in Tableau 0 were decreased sufficiently, then $u = 1$ would result by definition. Thus we wish to adjoin to the tableau a new equation (to be designated equation r) which is the same as eq (1) except that the coefficient of $-t_1$ is appropriately decreased. Such an equation can always be created by adding a sufficient positive multiple M of $t_1 = -1(-t_1)$ to eq (1). Defining $x_4 = x_1 + Mt_1 (\geqq 0)$ in Tableau 0, we have

$$x_4 = -98 - (1 + M)(-t_1) - 45(-t_2).$$

To assure $u = 1$ when equation 4 assumes the role of equation r, we require $-1/(1+M) > -28/45$ or $1 + M > 45/28$. Consequently we assign $-a_{14}$ the value $45/28 + \epsilon (-a_{14} = 1 + M)$, for $\epsilon > 0$.

Adjoining the new equation to Tableau 0 yields the augmented tableau.[21]

0′.

$x_0 =$	0	1	28
$x_1 =$	−98	−1	−45
$x_2 =$	0	−1	0
$x_3 =$	0	0	−1
$\rightarrow x_4 =$	−98	$-\left(\frac{45}{28} + \epsilon\right)$	−45

Note that by letting a_{14} be a_{ru} in this example, the process of selecting a value of a_{14} corresponds precisely to selecting a value of λ for a Gomory cut, whenever the transformation of A into $\overline{A}$ is carried out as specified in the PPDA.[22] In particular, $-a_{14} = \frac{45}{27}$ gives the (largest permissible) λ value prescribed by the All Integer Algorithm (using eq (1) as source equation).

Suppose instead, however, that we wish to select a_{14} by taking ϵ arbitrarily small. The effect of this in transforming A into $\overline{A}$ with the PPDA can easily be given without specifying a value of ϵ at all, provided the updated form of equation 4 itself is disregarded.

[20] One can readily make these conditions more precise by the type of "difference" analysis that permits the last two tableaus above to be inferred without carrying out the intervening computations. (There is of course no difficulty in defining (i) and (ii) for more general situations. For simplicity, however, we continue our discussion by reference to the example problem.)

[21] The transformations of the PPDA and the theorems of the preceding section do not require x_r to be an integer variable or equation r to contain integer coefficients.

[22] The Gomory all-integer algorithm can be described in terms of the algorithm of [1] in this way. See e.g., [2].

To see this we observe that for any number y and $-a_{ru}=y+\epsilon$ (with ϵ arbitrarily small), we have

$$[a_{rj}/(-a_{ru})]=[a_{rj}/y]-1$$

if $a_{rj}<0$ and a_{rj}/y is noninteger, while

$$[a_{rj}/(-a_{ru})]=[a_{rj}/y]$$

otherwise. Thus, by instruction 3 of the PPDA we obtain from Tableau 0′

1′.

			↓
→	−61	1	0
	−37	−1	−17
	61	−1	28
	0	0	−1

where we have dropped equation 4 from the new tableau.

However, dropping this equation is not permissible by a straightforward application of the PPDA since A is not lexicographically dual feasible, and equation r is required to define the transformation in Stage 2. To remedy this apparent difficulty, we note that Theorem 1 and its corollaries immediately imply that the equation $t_u=-1(-t_u)$ will be transformed by instruction 3 into an equation that satisfies the criteria for equation r.

In our present example, eq (3) corresponds to $t_u=-1(-t_u)$[23] in Tableau 0′ and hence qualifies as equation r in Tableau 1′. Applying instruction 5 of the PPDA to Tableau 1′ for $r=3$, we obtain

2′.

−61	1	0
−3	−18	17
5	27	−28
2	−1	1

The PPDA now obtains an optimal solution after six more steps, considerably improving upon the solution attempt that disregarded the form of the A matrix encountered in Stage 2 (as a result of the Stage 1 transformations).[24]

Other related ways for increasing the range of alternatives available to the PPDA are suggested by the foregoing discussion. For example, one may create a new equation to take the role of equation r by decreasing more than one of the a_{rj} (or even increasing a_{r0}), applying this in Stage 2 as well as Stage 1 provided $A_j \lfloor < 0 \rightarrow a_{rj}>0$ and $A_u^* \lfloor < A_s^*$ for the new equation r. Also, if there is a second equation that has the appropriate form for equation r in Stage 2 and has the same value for s, then it is easily proved that any convex combination of the two equations will qualify as the new equation r.

6. References

[1] Glover, Fred, A bound escalation method for the solution of integer linear programs, Cahiers du Centre L'Etudes de Recherche Operationnelle, **6,** Brussels (1964).

[2] Glover, Fred, An extension of the bound escalation method for integer programming: A pseudo primal-dual algorithm of the Gomory All-Integer variety, Management Sciences Research Report No. 49, Carnegie Institute of Technology (July 1965).

[3] Glover, Fred, A new foundation for a simplified primal integer programming algorithm, ORC 66–39, University of California, Berkeley (Nov. 1966), to appear in JORSA.

[4] Gomory, Ralph E., All-Integer Integer Programming Algorithm, Industrial Scheduling, J. F. Muth and G. L. Thompson, Eds. (Prentice Hall, 1963).

[5] Young, R. D., A primal (all-integer) integer programming algorithm, J. Res. NBS **69B** (Math. and Math. Phys.), No. 3, 213 (1965).

(Paper 71B4–243)

[23] If this equation does not appear in the tableau, it can always be added.

[24] Tableau 2′ illustrates the applicability of an additional solution strategy that can be employed in conjunction with the PPDA. The submatrix consisting of the two middle rows of the tableau is a special instance of a structure called the bounding form, which frequently appears in certain "hard" integer programs and can be exploited efficiently with the algorithm of [1].