Discussion Paper No. 16

ROW DROPPING PROCEDURES FOR CUTTING PLANE ALGORITHMS

by

Frederic H. Murphy

October 16, 1972

Graduate School of Management
Northwestern University
Evanston, Illinois

I wish to thank Professors Harvey Wagner, Gordon Bradley, and Robert Mifflin of Yale University for their helpful comments.

* This work was supported in part by National Science Foundation Grant GS - 3032.
Abstract

Row dropping procedures are provided for cutting plane algorithms. An example is provided to show that if nonbinding rows are dropped at every iteration while disregarding degeneracy in the linear program of the subproblem, the algorithms will not converge. Also, the results of Topkis [8] and Eaves and Zangwill [3] are reproduced using different techniques.
Topkis [8] and Eaves and Zangwill [1] have shown that constraints may be dropped at each iteration of certain cutting plane algorithms, provided that the objective function to be maximized is strictly quasi-concave. We show that the strict quasi-concavity condition can be modified to quasi-concavity in the case of the Veinott [9] algorithm and to concavity in the case of the Kelly [5] algorithm, provided that nonbinding rows are dropped only when the trial solution satisfies certain conditions. We also will show how to achieve the results in the strictly quasi-concave case by an alternative method of proof.

Consider the nonlinear programming problem

\[
\begin{align*}
\text{maximize} & \quad f(x_1, \ldots, x_n) \\
\text{subject to} & \quad g_i(x_1, \ldots, x_n) \leq 0 \quad \text{for } i = 1, \ldots, m \\
& \quad x_j \geq 0 \quad \text{for } j = 1, \ldots, n,
\end{align*}
\]

with \( f(\cdot) \) quasi-concave and \( g_i(\cdot), \ldots, g_m(\cdot) \) quasi-convex and \( g_i(0) < 0 \) for \( i = 1, \ldots, m \). To ensure that the origin is interior to the feasible region, we need only assume Slater’s constraint qualification and translate the coordinates of the \( x_j \)'s. Also, assume the feasible region is contained in a compact set \( X \), and that \( f(\cdot), g_i(\cdot), \ldots, g_m(\cdot) \) are continuous, finite and possess continuous gradients for all points in \( X \).

First we reformulate the problem as (NL2).
(2) \[ \text{maximize } x_0 \]

subject to

(3) \[ E_0(x) \equiv x_0 - f(x) - c \leq 0 \]

(4) \[ E_i(x) \leq 0 \text{ for } i = 1, \ldots, n \]

(5) \[ x_j \geq 0 \text{ for } j = 0, \ldots, n, \]

with \( c \) any constant such that \( f(0) + c > 0 \), and setting \( X = (x_0, x_1, \ldots, x_n) \).

we augment \( X \) appropriately to include the added dimension for \( x_0 \), and we define \( S \) as the feasible region to NLP.

Before we state what a cutting plane algorithm is, we define a central concept to cutting plane algorithms.

Definition of a Cutting Plane Function

In the manner of Torkis [8], we define a limiting cutting plane function.

**Definition 1**

A point to set mapping \((a(x), b(x))\) from \( X \to S \) into \( \mathbb{R}^{n+2} \) with \( a(x) \in \mathbb{R}^{n+1} \) and \( b(x) \in \mathbb{R} \) is a limiting cutting plane function if

(7) \[ S \subset H(x) \equiv \{ y: a(x)^T y \leq b(x) \} \]

for all \( x \in X \cap S \), \((a(x), b(x))\) is bounded on \( X \to S \), and, for any \(\{x_k \mid k = 1, 2, \ldots\}\), with \( \lim_{k \to \infty} x_k = \bar{x} \in X \cap S \) the limit point \((\bar{a}, \bar{b})\) of any convergent subsequence \(\{a(x_k), b(x_k)\}\) satisfies \(\bar{a} \cdot \bar{x} \leq \bar{b}\).

Let \( G(x) = \max \{ E_0(x), \ldots, E_m(x) \} \) and \( u(x) \) be the gradient of a
If we assume each $g_i(x)$ is convex, then $G(x)$ is a convex function, and we have

$$G(y) \geq G(x) + u(x) \cdot (y-x)$$

for all $x, y \in X$. Topkis [8] shows that

$$[u(x), u(x) \cdot x = G(x)]$$

is a limiting cutting plane function, which is the one used by Kelley [5]. Note that convexity is important. Without convexity, if, for some $y$, $G(y) \leq 0$ and $G(x) + u(x) \cdot (y-x) > 0$, then part of the feasible region might be cut off at some iteration. By (8) we are assured that this will not happen.

We have assumed that $G(0) < 0$. For $x \in X \cap S$, define

$$\lambda(x) = \max \{\lambda : \lambda x \in S\}$$

and set $w(x) = a(x)x$ for any function $a(x) \in [\lambda(x), 1]$ with $\frac{\partial w(x)}{\partial x} \in [0, 1]$. Topkis [8] shows that

$$[u(w(x)), u(w(x)) \cdot w(x) = G(w(x))]$$

is a family of cutting plane functions. By setting $a(x) = \lambda(x)$, (10) becomes the cutting plane function proposed by Veinott [9]. For using Veinott's cutting plane function, the appropriate assumption on the constraints is quasi-convexity with $\forall g_i(x) \neq 0$ when $g_i(x) = 0$. Otherwise, we need the assumption of convexity to use (10) when $\lambda(x) \leq a(x) \leq \lambda$. Note that (9) is the special case of (10) where $a(x) = 1$.

The Algorithm

Assume that $X$ is a bounded convex polyhedron and, for convenience,
that we can specify $0 < b_j < \infty$ such that we can let $X = \{x|b_j \geq x_j \geq 0 \text{ for } j = 0,\ldots,n\}$. At iteration $k$ we have a linear program (LPI) of the form

$$\text{maximize } y_0 \equiv p_k$$
subject to

$$\sum_{j=0}^{n} a_j^i(x_k^i)x_j - b(x_k^i) \leq 0 \text{ for } i = 1,\ldots,r_k$$

$$b_j \geq x_j \geq 0 \text{ for } j = 0,\ldots,n,$$

where the vectors $x_k^i$ are solutions from previous iterations of LPI (at iteration 0 LPI consists of (11) and (13)).

For the algorithm choose either the cutting plane function in (9) or one from the family in (10). Evaluating the cutting plane function at $x_k^i$, we have $a_0(x_k^i),\ldots,a_n(x_k^i)$, and $b(x_k^i)$, the coefficients in LPI. Also, $r_k$ is the number of constraints in the linear program at iteration $k$, and $b_j$ is the bound on the $j^{th}$ variable in the definition of $X$.

The dual of LPI (DLP) is

$$\text{minimize } \sum_{i=1}^{r_k} b(x_k^i)y_i + \sum_{j=0}^{n} b_jz_j$$
subject to

$$\sum_{i=1}^{r_k} a_i^j(x_k^i)y_i + x_j - s_j = 1$$

$$\sum_{i=1}^{r_k} a_i^j(x_k^i)y_i + x_j - s_j = 0 \text{ for } j = 1,\ldots,n.$$ 

$y_i, z_j, s_j \geq 0$ for $i = 1,\ldots,r_k$ and $j = 0,\ldots,n$.

with the values $k_i^k$, for $i = 1,\ldots,r_k$, and $k_j^k$ and $s_j^k$ for $j = 0,\ldots,n$, given by an optimal basic solution.
Solve DLPI. If \( X_k \) is feasible, \( X_k \) is optimal and we stop. Otherwise, we proceed as follows. If Condition 1 below is satisfied, we drop all nonbasic columns not associated with the \( z_j \) and add a new column to DLPI determined by evaluating the cutting plane function at \( X_k \). The optimal solution to LPI is associated with our solution to DLPI, setting \( X_{k+1} = X_k \).

**Condition 1**

(a) In DLPI \( y_i^k \leq \epsilon, z_j^k \geq \epsilon \) and \( s_j^k \geq \epsilon \) for all basic \( y_i^k, z_j^k \), and \( s_j^k \).

(b) The absolute value of the determinant of the basis is greater than \( \epsilon \), where \( \epsilon > 0 \) and fixed for all iterations.

If Condition 1 is violated, our iteration consists of adding the new column to DLPI without dropping any nonbasic columns. As soon as Condition 1 is again satisfied we resume dropping nonbasic cutting planes. We show that if \( X_k \) is not feasible in NLP at some finite iteration then any convergent subsequence of \( X_k \) converges to an optimal solution of NLP.

The following example illustrates the need to avoid degeneracy in DLPI, which corresponds to multiple optima in LPI.

(17) \[
\text{maximize } x_1 + x_2
\]

subject to

(18) \[
(x_1 - c_1)^2 + (x_2 - c_2)^2 \leq c_3
\]
Because (17) is a linear function we need not reformulate the nonlinear program in the form (4), (5) and (6). This allows the following diagram to be two dimensional instead of three dimensional to better illustrate the degeneracy problem. Therefore, let LPI at iteration 1 be

(20) \begin{align*}
& \text{maximize } x_1 + x_2 \\
\text{subject to} \\
& x_1 + x_2 \leq b^1 \\
& x_1 \leq b_1 \\
& x_2 \leq b_2 \\
& x_1, x_2 \geq 0, 
\end{align*}

where $b^1 = b(x(0))$.

Pictorially we have
The cross-hatched area is the feasible region of NLP; and the line connecting $X_{[1]}$ and $x$ is determined by (21). The feasible region of (20),···,(24) is the portion of the rectangle to the left of this line. Observe that all points between $x$ and $X_{[1]}$ are alternative optima in LP. Suppose that the point $X_{[1]}$ is the solution produced by the simplex algorithm. The cutting plane provides a boundary, $l$, to the new linear programming feasible region that is perpendicular to the line connecting $X_{[1]}$ and $(c_1,c_2)$, by the symmetry of $(x_1-c_1)^2 + (x_2-c_2)^2$. The portion of the feasible region defined by (21),···,(24) above $l$ is now cut off.

If the optimal solution at iteration 2 is $X_{[2]} = x$, dropping all non-binding constraints makes $X_{[1]}$ feasible again, allowing the algorithm to alternate between $X_{[1]}$ and $x$ as trial solutions, never converging to an optimal solution to NLP.

Part (b) of Condition 1 is used to keep the constraint matrix coefficients uniformly bounded in DLPI at an optimal solution. To see this, first note that the coefficients in DLPI are uniformly bounded because they are determined by continuous functions evaluated at points in a compact set. To get to the optimal simplex tableau of DLPI, we perform multiplications and additions involving the uniformly bounded coefficients of DLPI and the inverse of our optimal basis. That the elements of inverse matrix are uniformly bounded is seen by calculating the inverse by using the matrix of cofactors, whose elements are bounded, and dividing by the determinant of the basis matrix which is bounded away from zero by (b).

Condition 1 in theory is stronger than a condition that allows column dropping whenever there are no multiple optima in LP. However, from a computational point of view the two conditions are equivalent. If we set $ε$ at the closest number to zero that the computer can handle, then
part (a) of Condition 1 translates into "degeneracy" within numerical tolerances of the computer. Part (b) of Condition 1 becomes a requirement that the coefficients of DLPI are within the capability of the computer to handle while performing simplex iterations.

If the simplex or any related algorithm is not used to solve DLPI and the practical issue of coefficients increasing in an unbounded fashion due to linear dependence is not faced, we can provide an altered version of Condition 1.

**Condition 2**

(a) In DLPI \( y_i^k \geq \epsilon, \quad z_j^k \geq \epsilon \) and \( s_i^k \geq \epsilon \) for all basic \( y_i^k, z_j^k, \) and \( s_i^k, \) where \( \epsilon > 0 \) and fixed for all iterations.

(b) Once (a) is violated, columns are no longer dropped from DLPI.

Either Condition 2 is violated at some finite iteration and convergence is guaranteed by the proofs in Torkis [8] or Condition 2 is satisfied at every iteration and convergence is guaranteed by algorithm IV in [3].

There is still another condition which allows us to drop columns from DLPI.

**Condition 3**

In DLPI, start with the optimal solution determined at iteration \( k-1 \) as the initial trial solution at iteration \( k \). If the new column added by the cutting plane function can be pivoted into the basis at a level at least \( \epsilon > 0 \), \( \epsilon \) fixed for all iterations, then all nonbasic columns in DLPI associated with the solution \( y_i^{k-1}, z_j^{k-1}, s_i^{k-1} \) for \( i = 1, \ldots, r_k \) and \( j = 0, \ldots, n \), retained from iteration \( k-1 \) may be dropped.
Let

\[
\bar{x}_k = \begin{cases} 
    \bar{x}_{k-1} & \text{if } \sum_{i} g_i(\bar{x}_k) < 0, g_i(x_k) > 0 \text{ for some } i \\
    x_{[k]} & \text{if } \sum_{i} g_i(\bar{x}_k) > 0, g_i(x_{[k]}) > 0 \text{ for some } i
\end{cases}
\]

where \( \bar{x}_1 = x_1 \). For this condition we will show that any convergent subsequence of \( \bar{x}_k \) is an optimal solution to NLP if the algorithm does not terminate finitely. To apply Condition 1 as it is stated, we have to start the succeeding iteration before we know whether we can drop nonbasic columns from the present iteration. Also, all we show is the existence of a subsequence converging to an optimal solution of NLP. This condition is, however, a generalization of Condition 1. If Condition 1 holds, since all coefficients are uniformly bounded and we have a basic solution uniformly bounded away from zero, we can pivot in our new column at a level greater than or equal to some fixed number greater than zero.

**Convergence Using Condition 1**

If there is no uniform bound on the number of columns in NLPI, for every iteration, then convergence is guaranteed by the original convergence proofs. Therefore, we need only consider the case where there is a uniform bound on the number of columns in NLPI.

For a sequence of points \( x_k \in X \) with \( x_k \neq x \) let \( g_h(x_k) = G(x_k) \) for some fixed \( h \). By construction \( G(x) \) is continuous. Consequently, \( g_h(x) = G(x) \). Now,

\[
h_k(x) = \varphi g_h(x_k) + (x - x_k) + g_h(x_k)
\]

is a linear equation that defines a hyperplane tangent to \( g_h(x) \) at \( x_k \) and is an appropriate cutting plane at \( x_k \). By the continuity of the gradient of \( g_h(x) \),
(27) \[ \lim_{k \to \infty} \nabla g_h(x_k) = \nabla g_h (\bar{x}), \]

that is:

Lemma 1

(28) \[ h(x) = \nabla g_h(\bar{x}) \cdot (x-\bar{x}) + g_h(\bar{x}) \]

defines a tangent hyperplane to \( g_h(x) \) at \( \bar{x} \) and is an appropriate cutting plane at \( \bar{x} \).

Lemma 2

For every \( h(x) = \sum_{j=0}^{n} a_j x_j - b \) that defines a tangent hyperplane to a constraint function \( g_4(x) \) at an \( \bar{x} \in X \), if

(a) \( g_4(x) \) is convex, or

(b) \( g_4(x) \) is quasi-convex and for \( \bar{R} \) with \( g_4(\bar{R}) = 0 \) \( \forall \bar{R} \neq 0 \),

then there is a uniform bound \( \delta > 0 \) such that \( b \geq \delta \) for all \( \bar{x} \in X \).

Proof: By assumptions (a) or (b) and since \( 0 \preceq G(0), b \preceq 0 \). Assuming the convexity of \( g_4(x) \),

(29) \[ g_4(x) \preceq G(x) \geq g_4(x) > \sum_{j=0}^{n} a_j x_j - b, \]

(30) \[ 0 \preceq G(0) \geq -b. \]

Setting \( G(0) = -b \) we have proved the lemma under hypothesis (a).

Let us assume for all \( \delta > 0 \) there is an \( x^\delta \) satisfying (b) for \( g_4(x) \)

with \( h_\delta(x) = \sum_{j=0}^{n} a_j x_j - b^\delta \), where \( (b^\delta, \ldots, b^\delta) = \nabla g_4(x^\delta) \), having \( 0 \leq b^\delta < \delta \).

Let \( \bar{x} \) be the limit of a sequence of \( x^{\delta} \)'s for a sequence of \( \delta \)'s converging to zero. Note that by assumption \( g_4(\bar{R}) = 0 \), and so \( \bar{x} \neq 0 \). Because \( \delta \to 0 \),
(31) \[ \sum_{j=0}^{n} a_j x_j^i \leq \sum_{j=0}^{n} \bar{x}_j^i = 0 \text{ as } \delta \to 0, \]

with \((\bar{x}_0, \ldots, \bar{x}_n) = \nabla g_i(x)\).

Since \(g_i(x)\) is pseudo-convex, \(g_i(0) < g_i(x)\) implies \(\sum_{j=0}^{n} \bar{x}_j^i (0 - \bar{x}_j^i) < 0\),

which contradicts (31).

Therefore, there is a \(\delta > 0\) such that

(32) \[ b > \delta. \]

Now we can bound the dual variables in DLPI.

Lemma 3 In DLPI \(y_i^k\) and \(z_j^k\) are uniformly bounded for all \(i\) and \(j\) and \(k\).

Proof:

(33) \[ b(x_j^k) \geq \delta > 0 \]

and with \(b_0\) one of the upper bounds defining \(X\)

(34) \[ b_0 \geq \bar{x}_0, \]

or, since \(p_k\) is the optimal value of LP1,

(35) \[ b_0 \geq p_k, \]

which means that

(36) \[ b_0 \geq \frac{r_k}{\kappa} + \sum_{i=1}^{\kappa} b(x_i^k) y_i^k + \sum_{j=0}^{n} b_j^k z_j^k. \]

Since \(b(x_i^k) > 0\) for all \(i\) and \(b_j^k > 0\) for all \(j\),

(37) \[ b_0 \geq b(x_j^k) y_j^k \text{ for } i = 1, \ldots, r_k. \]
Therefore.

\[ x_i^k \leq \frac{b_i}{b} \quad \text{for } i = 1, \ldots, r_k. \]

Similarly

\[ x_j^0 \leq \frac{b_0}{b_j} \quad \text{for } j = 0, \ldots, n. \]

Before continuing, we clarify a point about the construction of DLPL.

At each iteration we drop all nonbasic columns associated with the constraints in (12). After doing that, we reindex all our columns that remain retaining the columns in the same order from left to right and add our new column to the rightmost position in the portion of the matrix for columns associated with the constraints in (12). Doing this means that the \( y_k^i \) that determines column \( i \) at iteration \( k \) is different from the \( y_{k+1}^i \) at iteration \( k+1 \) if for some \( h \leq i \) the column determined by \( y_k^h \) is dropped.

The reason for making this clear is that we shall use a subsequence on which the \( y_k^i \)‘s that determine column \( i \) on this subsequence will converge to a limit. All that we need to know about these \( y_k^i \)‘s for the subsequence to exist is that they are in the compact space \( X \). We don’t need to know anything as to how they came to determine column \( i \) for the purpose of taking a subsequence.

By the definition of limiting cutting plane function, \( a_j(x_k^i) \) for \( b(x_k^i) \) are bounded for all \( i, j \) and \( k \). Therefore we may take a subsequence indexed by \( k \) with the following properties:

1. Condition 1 is satisfied
2. \( a_j(x_k^i) \to a_{ij} \) for \( i = 1, \ldots, r \) and \( j = 0, \ldots, n \)
(3) \( b(k_w^i) \leq b_m^i \) for \( i = 1, \ldots, r \)

(4) \( X[k_w] \leq X \)

(5) The indices of the basic columns in DLPI corresponding to constraints (12) in LPI at \( k_w \) are the same for all \( k_w \), with the number of these columns being \( r' - 1 \).

(6) \( a_j(X[k_w]) = a_{r+1}^j \) for \( j = 0, \ldots, n \)

(7) \( b(X[k_w]) = b_m \)

(8) \( y_i^w = y_i^w \) and \( z_j^w = z_j^w \) for all \( i \) and \( j \).

We may now construct a limiting linear program (LLPL) to the subsequence indexed by \( k_w \):

\[
\begin{align*}
\text{maximize } & \quad x_0 = p_m \\
\text{subject to } & \quad \sum_{j=0}^{n} a_j^{w} x_j^{w} + b_m^i \leq 0 \text{ for } i = 1, \ldots, r \\
& \quad b_j \geq x_j \geq 0 \text{ for } j = 0, \ldots, n.
\end{align*}
\]

Since \( p_k \) is monotonically decreasing and bounded from below by \( f(0) + c \), we have \( \{p_k\} \) converging to some limit \( \bar{p} \). By the Hoffman and Karp [4] result on the continuity of linear programs, we conclude \( p_m = \bar{p} \).

As in Murphy [5], we construct a limiting linear program for the successor subsequence [10], that is, the subsequence of iterations indexed by \( k_w + 1 \). Let \( |Q| = r' - 1 \). Then the number of columns in DLPI at iteration \( k_w + 1 \) is \( r' + n + 1 \), since we save active columns in DLPI.
corresponding to constraints (12) in LPI, all active and inactive columns corresponding to upper bounds, and add a new column. In LPI constraint \( i \) at iteration \( k^+ \) is constraint \( \ell(i) \) at iteration \( k^w \).

We have shown that the coefficients of the constraint added for iteration \( k^w+1 \) converge to a limit, and the constraints \( \ell(i) \) converge by the construction of \( k^w \). Noting lemma 1, we can perform a cutting plane iteration on LPI using the limits of the coefficients of the constraints added at iteration \( k^w+1 \). Also, nonbinding constraints are dropped since the limiting solution to DLPI satisfies the row dropping rule. Therefore, we arrive at LLP2 as a limit of a subsequence of linear programs and as a cutting plane iteration to LPI where LLP2 is:

\[
\text{maximize } x_0
\]

subject to

\[
\sum_{j=0}^{n} b^i_{j+1} x_j - b^i_0 \leq 0 \quad \text{for } i = 1, \ldots, r'
\]

\[
b_j x_j \geq 0 \quad \text{for } j = 0, \ldots, n.
\]

The value of the optimal LLP2 is also \( p_x \) since any subsequence of the convergent sequence \( p_x \) has the same limit and because of the continuity of linear programs [4].

**Lemma 4** Let \( y^+_{i+1} = y^m_{\ell(i)} \) for \( i = 1, \ldots, r' - 1 \) and \( y^+_{r'} = 0 \). Then \( y^+_{i} \), for \( i = 1, \ldots, r' \), is an optimal solution to the dual of LLP2 with
\[ x_j^{m+1} = x_j^m \] for all \( j \).

**Proof:** Note that \( y_i^{m+1} \), for \( i = 1, \ldots, r' \), is a feasible nondegenerate solution to the dual of LLP2 since, \( y_i^m \) for \( i = 1, \ldots, r \), is feasible in the dual of LLP1. Also, \( y_i^m \), for \( i = 1, \ldots, r' \), is optimal because the optimal value of LLP2 is \( p_\omega \), and this solution has the value \( p_\omega \) in the dual of LLP2.

Therefore, we have:

**Lemma 5** The point \( x_m^\star \) is feasible in NLP.

Look at the optimal simplex tableau of the dual of LLP1 using the solution \( y_1^{m}, \ldots, y_r^{m}, z_1^{m}, \ldots, z_n^{m} \). The components of \( x_m^\star \) are the coefficients of the slack variables in the objective function. Using the cutting-plane function, we determine a new column to be added to the dual of LLP1. Dropping all nonbasic columns still leaves the solution to the dual of LLP1 feasible.

Since the dual of LLP2 has the same value for an optimal solution, \( p_\omega \), our optimal nondegenerate basic feasible solution to the dual of LLP1 is optimal in the dual of LLP2. Since the solution is nondegenerate, if the added column has a negative relative price in the simplex tableau [1], the value of the optimal solution would decrease by pivoting in the new column, contradicting the Hoffman and Karp result [4]. This means the associated primal solution to LLP1 is feasible in LLP2; that is, \( x_m^\star \) is feasible and optimal in LLP2. Or, since \( x_m^\star \) is not cut off by our cutting plane function, it is feasible in NLP.
By the construction of the subsequence \( k_v \), we see that any convergent subsequence of \( X_{k_i'} \)'s is feasible in NLP as long as it is a subsequence where Condition 1 is always satisfied. Let \( k_v \) index any convergent subsequence of \( X_{k_i'} \)'s with \( X_{k_v} \rightarrow \tilde{X} \) and \( k_z \) index the last iteration before iteration \( k_v \) where cutting planes are dropped. We are still assuming that Condition 1 is satisfied infinitely often, so that \( [k_z] \) is infinite.

Taking an appropriate convergent subsequence of \( X_{k_z} \) without reindexing, we have \( X_{k_z} \rightarrow X_m \), and a limiting linear program (LLP3) can be constructed as before from the subsequence \( k_z \). We take the appropriate convergent subsequence of \( X_{k_v} \), again without reindexing, so that with our new subsequence the index \( k_z \) is still the last iteration where cutting planes are dropped before iteration \( k_v \). Now \( X_{k_v} \) is feasible in LPI at iteration \( k_z \) and \( p_{k_z} \leq p_{k_v} \leq p_m \). Therefore, by taking limits, \( \tilde{X} \) is a feasible and optimal solution to LLP3; and \( \tilde{X} = X_m \) by Condition 1, which insures there are no multiple optima to LLP3.

Noting that convergence is guaranteed by the proofs of convergence in Topkis [8] when there is no uniform bound on the number of constraints in LPI, we can say

**Theorem 1.** The limit point of any convergent subsequence of optimal solutions to LPI is feasible in NLP, and, therefore, optimal.

**Proof of Convergence using Condition 3.**

Again, as with Condition 1 we need only concern ourselves with the case where there is a uniform bound on the number of columns in OLPI.
Lemma 6. Let $X_u$ be an infinite subsequence of iterations where
Condition 3 is satisfied and $X_u^*$ is feasible. Then $X_u$ is a feasible solution
to NLP.

Proof: Let

$$d_u = \sum_{y=0}^{\infty} a(X_u^y)X_u^y - b(X_u^y).$$

Assume we can take a subsequence of $X_u^y$, retaining our index $k_u$, such that

$$\lim_{k_u} d_u \geq 2y > 0.$$ 

For (47) to hold, by the definition of limit, there exists a $K$ such
that $d_u \geq y$ for $k_u \geq K$. But $d_u$ is the coefficient in the objective
function of the new column in DLPI relative to the basis retained from
iteration $k_u - 1$ in DLPI at iteration $k_u$. Pivoting into the basis this new
column at a level greater than $c$ decreases the objective function by an
amount greater than $d_u \cdot c$, or $c \cdot \varepsilon$. Having an improvement of $\gamma \cdot \varepsilon$
infinately often means the values of the optimal solutions to DLPI are
unbounded below, because the values of the optimal solutions to the DLPI's are
monotonically decreasing with the decrease at iterations $k_u$ greater
than $\varepsilon$. This contradicts the existence of a feasible solution to NLP,
$k_u \to 0$, or $X_u$ is feasible in DLPI.
Theorem 2. Any convergent subsequence of $x_k$ as defined in (25) is optimal in NLP.

Proof: Note that

\[ \sum_{i} \frac{E_1(x_k)}{E_1(x_{k+1})} \leq \sum_{i} \frac{E_1(x_k)}{E_1(x_{k+1})}. \]

By the definition of $x_k$

\[ \sum_{i} \frac{E_1(x_k)}{E_1(x_{k+1})} \leq \sum_{i} \frac{E_1(x_{k+1})}{E_1(x_{k+1})}. \]

By lemma 6,

\[ \sum_{i} \frac{E_1(x_{k+1})}{E_1(x_{k+1})} \rightarrow 0. \]

Therefore, by (48), (49), and (50), we have

\[ \sum_{i} \frac{E_1(x_k)}{E_1(x_{k+1})} \rightarrow 0, \]

or, any convergent subsequence of $x_k$ is feasible in the limit and the theorem holds.

Convergence with a Strictly Quasi-concave Objective Function

The following is an alternative method of proof of the results of Topkis [8] using the concept of the continuity of mathematical programs [7].
Here we solve a sequence of mathematical programs consisting of a nonlinear objective function and linear constraints (MPl)

\[
\begin{align*}
\text{maximize } & \quad f(x) = p_k \\
\text{subject to } & \quad \sum_{j=0}^{n} a_j(x_k^i)x_j - b(x_k^i) \leq 0 \text{ for } i = 1, \ldots, r \quad \text{(53)} \\
& \quad b_j \geq x_j \geq 0 \text{ for } j = 0, \ldots, n. \quad \text{(54)}
\end{align*}
\]

Let \( x[k] \) be an optimal solution to MPl at iteration \( k \). At each iteration we use a cutting-plane function to add a constraint to cut off the optimal solution to MPl. At the same time we drop all nonbinding constraints (53) in MPl.

Lemma 7 The value of the optimal solution of MPl is monotonically decreasing, that is, \( p_k > p_{k+1} \) if \( x[k] \) is not feasible in NLP.

Proof: Since \( x[k] \) is the optimal solution to MPl at iteration \( k \), \( x[k] \) is still the optimal solution to MPl with nonbinding constraints removed. By adding the new cut to MPl with nonbinding constraints removed, the feasible region is reduced in size. Therefore, \( p_k > p_{k+1} \).

Assume \( p_k = p_{k+1} \). By strict quasi concavity

\[
\begin{align*}
p_k & < f(\lambda x[k] + (1-\lambda)x[k+1]) \quad \text{for } 0 < \lambda < 1. \quad \text{(55)}
\end{align*}
\]

Since only a finite number of nonbinding constraints are dropped at any iteration, there is a \( \lambda^k < 1 \) such that \( \lambda^k x[k] + (1-\lambda^k)x[k+1] \) is feasible in MPl at iteration \( k \). Using (54), this contradicts the optimality of \( x[k] \) in MPl at iteration \( k \), which means \( p_k > p_{k+1} \).
Now we must investigate the continuity of nonlinear programs of the form \( MP_1 \). This issue is treated with more generality in [2]. First we introduce new notation. Let \( T_k \) be a subsequence of sets. We define

\[
\lim T_k = \{ x | x_k \to x \text{ for } x_k \in T_k \}
\]

and

\[
\overline{\lim} T_k = \{ x | x_k \to x \text{ for } x_k \in T_k \}.
\]

The \( \lim \inf \) of \( T_k \), \( \lim T_k \), is the set of limits of sequences and the \( \lim \sup \) of \( T_k \), \( \overline{\lim} T_k \), is the set of limits of subsequences. This means

\[ \lim T_k \subset \lim \inf T_k \subset \lim \sup T_k. \]

If \( \lim T_k = \overline{\lim} T_k \), then the limit of a sequence of sets is defined as

\[
\lim T_k = \lim T_k = \overline{\lim} T_k.
\]

Lemma 8 \ Let \( A_k \) be a sequence of \( m \times n \) matrices with \( A_k \to A \) and \( b_k \) be a sequence of \( m \) dimensional vectors such that \( b_k \to b \). Let \( X \) be a compact set. Also, let \( T_k = \{ x | A_k x \leq b_k, x \in X \} \) and \( T = \{ x | A x \leq b, x \in X \} \). Assume there exists an \( x_0 \) with \( A_k x_0 \leq b_k + \delta \begin{pmatrix} 1 \\ 1 \end{pmatrix} \) with \( \delta > 0 \) for all \( k \). Then

\[ \lim T_k = T. \]

Proof: \ Let \( x \) be in \( T \), so that \( a x \leq b \). Since \( A_k \to A \), \( b_k \to b \), and the components of \( x \) are bounded, because \( x \in X \), then for all \( \epsilon > 0 \), there exists \( K_\epsilon \) such that

\[
A_k x \leq b_k + \epsilon_k \begin{pmatrix} 1 \\ 1 \end{pmatrix} \text{ for } k \geq K_\epsilon.
\]
where $\epsilon_k \geq 0$ and $\epsilon_k \to 0$. This means

\begin{equation}
A_k \left( (1 - \frac{\epsilon_k}{\delta}) x + \frac{\epsilon_k}{\delta} x_0 \right) \leq (1 - \frac{\epsilon_k}{\delta}) b_k + (1 - \frac{\epsilon_k}{\delta}) \begin{pmatrix} 1 \\ l \end{pmatrix} + \frac{\epsilon_k}{\delta} b_k = \frac{\epsilon_k}{\delta} \begin{pmatrix} 1 \\ l \end{pmatrix} \leq b_k
\end{equation}

Therefore, $(1 - \frac{\epsilon_k}{\delta}) x + \frac{\epsilon_k}{\delta} x_0$ is feasible in $T_k$ and converges to $x$, which implies $T \subseteq \lim T_k$.

Now let $x_{k_w} \in T_{k_w}$ and $\lim_{k_w \to \infty} x_{k_w} = \bar{x}$. Since $X$ is compact, $\bar{x} \in X$.

Since $x_{k_w} \in T_{k_w}$, $A_{k_w} x_{k_w} \leq b_{k_w}$ with the result that

\begin{equation}
\lim_{k_w \to \infty} (A_{k_w} x_{k_w}) \leq \lim_{k_w \to \infty} b_{k_w} = b.
\end{equation}

by Rudin [7]

\begin{equation}
\lim_{k_w \to \infty} (A_{k_w} x_{k_w}) \leq \lim_{k_w \to \infty} b_{k_w} = b,
\end{equation}

that is,

\begin{equation}
A \bar{x} \leq b,
\end{equation}

or

\begin{equation}
T \subseteq \lim T_k.
\end{equation}

Therefore we have

\begin{equation}
T = \lim T_k.
\end{equation}
Theorem 3: Assume we have a sequence of mathematical programs

\[ \begin{align*}
\text{(66)} & \quad \max_{x \in X} c_k(x) = p_k \\
\text{(67)} & \quad A_kx \leq b_k
\end{align*} \]

subject to

with nonempty feasible regions, \( c_k(x) = c(x) \), a continuous function for \( x \in X \), where \( X \) is a compact space, \( A_k \to A \), and \( b_k \to b \), consider the mathematical program

\[ \begin{align*}
\text{(68)} & \quad \max_{x \in X} c(x) = p \\
\text{(69)} & \quad Ax \leq b.
\end{align*} \]

Then \( \lim p_k = p \), if there exists an \( x_0 \) with \( A_kx_0 \leq b_k - \delta \)

\[ \begin{pmatrix} 1 \\ 1 \end{pmatrix} \]

with \( \delta > 0 \) for all \( k \).

Proof: Using Lemma 8 and setting \( T_k = \{ x | A_kx \leq b_k, x \in X \} \) and

\( T = \{ x | Ax \leq b, x \in X \} \), we have \( \lim_{k \to \infty} T_k = T \). Let \( Y_k \) be the set of optimal solutions to (66) and (67) and let \( Y \) be the set of optimal solutions to (68) and (69). For every \( \epsilon > 0 \), there is a \( K_\epsilon \) such that

for \( k \geq K_\epsilon \) and any \( x \in T \), there exists an \( x_k \in T_k \) where \( |x_k - x| < \epsilon \)

and, for any \( x_k \in T_k \), there exists an \( x \in T \) where \( |x_k - x| < \epsilon \), and
\[ |c(x) - c_k(x)| < \varepsilon \text{ for } x \in X. \text{ For } X_{[k]} \in Y_k \text{ we have} \]
\[ |c(X_{[k]}) - c_k(X_{[k]})| < \varepsilon \text{ for } k \geq k_\varepsilon, \]
and there exists an \( x \in X \) with
\[ |X_{[k]} - x| < \varepsilon \text{ for } k \geq k_\varepsilon. \]
By the continuity of \( c(x) \) on \( X \),
\[ |c(X_{[k]}) - c(x)| < \varepsilon + \lambda, \]
where \( \lambda \to 0 \text{ as } \varepsilon \to 0. \]
Since \( c(X_{[k]}) = p_k \) and \( p \geq c(x) \),
\[ p_k \leq p + \varepsilon + \lambda \text{ for } k \geq k_\varepsilon. \]
In like manner,
\[ p \leq p_k + \varepsilon + \lambda \text{ for } k \geq k_\varepsilon. \]
Hence
\[ |p - p_k| \to 0 \text{ as } k \to \infty, \]
completing the proof.

We have here the same behavior as before with the limiting linear programs. We want to construct a subsequence of MPl's that converge to a limiting mathematical program. Therefore, we take a subsequence, indexed by \( k_\mu \), with the following properties:
(1) \( a^i_{x^i} \rightarrow a^i_{x^i} \) for \( i = 1, \ldots, r \), and \( j = 0, \ldots, n \)

(2) \( b(x^i_{x^i}) \rightarrow b^i_{x^i} \) for \( i = 1, \ldots, r \)

(3) \( X_{[k_w]} \rightarrow X_w \)

(4) the binding constraints in MPL are the same for all \( k_w \),
   with the number of these being \( r' + 1 \)

(5) \( a^i_{j}(X_{[k_w]}) \rightarrow a^i_{r'} \)

(6) \( b(X_{[k_w]}) \rightarrow b^i_{x^w} \).

We may now construct a limiting mathematical program (LMP1) to the
subsequence indexed by \( k_w \):

(76) \[
\begin{array}{c}
\text{maximize} \\
\quad c(x) = a^i
\end{array}
\]
\[ x \in X \]

subject to

(77) \[
\sum_{j=0}^{n} a^i_{ij} x_j - b^i_{x^w} \leq 0 \quad \text{for } i = 1, \ldots, r.
\]

By using Lemma 2 we see that \( A_k \cdot \delta = b^i \cdot \delta \) \( \delta = 0 \) and the conditions of

Theorem 1 are satisfied. With a proof similar to that of Lemma 4, we can show
that \( X_\infty \) is an optimal solution to LMP1. Performing a cutting-plane iteration
on LMP1, we arrive at a second mathematical program (LMP2)

(78) \[
\begin{array}{c}
\text{maximize} \\
\quad c(x)
\end{array}
\]
\[ x \in X \]

subject to

(79) \[
\sum_{j=0}^{n} a^i_{ij} x_j - b^i_{x^w} \leq 0 \quad \text{for } i = 1, \ldots, r'.
\]
where \( x_{(1)} = a \) and where constraint 1 at iteration \( k+1 \) is constraint \( z(1) \) at iteration \( k \). Note also that LMP2 is the limiting mathematical program to the subsequence \( k+1 \). Since the value of an optimal solution, \( p_k \), is monotonically decreasing by Theorem 3 both LMP1 and LMP2 have the same optimal solution values, \( p \), which leads us to

**Theorem 4.** The point \( X_k \) is an optimal solution to NLP.

**Proof:** Assume \( X_k \) is not optimal in NLP. By Lemma 7 the value of an optimal solution to LMP2 should be strictly less than \( p \) if \( X_k \) is not optimal in NLP, contradicting the fact that \( p \) is the value of an optimal solution in LMP2.
References


