# A NEW FACE METHOD FOR LINEAR PROGRAMMING

PING-QI PAN

Abstract. An attractive feature of the face method [9] for solving LP problems is that it uses the orthogonal projection of the negative objective gradient on the related null space as the search direction. However, the method would not be amenable for solving large sparse problems, since it handles the involved normal system by orthogonal transformations. This work derives a new type of search directions using Gaussian elimination, and formulates an algorithm using LU factorization. In a single iteration, the objective reduction by the latter is of the squares order, compared with the first order reduction by the simplex algorithm. The search direction is shown to be the best in certain sense. Some anti-degenerate tactics are incorporated by taking advantage of the face framework.

## 1. Introduction

We are concerned with the standard linear programming (LP) problem

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & Ax = b, \quad x \geq 0, \end{aligned} \tag{1.1}$$

where $A \in R^{m\times n}$ $(m < n)$, $b \in R^m$, $c \in R^n$; $\operatorname{rank}(A) = m$.

In the underlining polyhedron, the simplex method [2] moves from vertex to vertex, until reaching an optimal vertex, whereas the interior point method moves from interior point to interior point, until arriving in a sufficiently small neighborhood of an optimal point. These methods would be classified into *point* methods. The deficient basis simplex method [4, 7] and the *projective* method [5, 6, 8] also fall into this category. But the latter is the predecessor of the face method [9], which might be viewed as a *set* method — it goes from face to face until reaching an optimal face together with a pair of primal and dual optimal solutions. One of its attractive features is that it applies an "ideal" search direction, that is, the orthogonal projection of the negative objective gradient onto the related null space. Moreover, in each iteration, it solves only two triangular systems, compared with two standard systems in the simplex method. Numerical results obtained from preliminary computational experiments are quite favorable.

Nevertheless, the face method handles the involved normal system by the orthogonal transformation, or alternatively by inversion of system's coefficient matrix [11], as would not be amenable for solving large sparse LP problems. A variant of the face method was proposed to overcome this weakness, but it turned out to be unstable [10, 13].

In this paper, a least squares problem is handled to create a new type of search directions using the Gaussian elimination. An algorithm is then formulated using the LU factorization. In a single iteration, the objective reduction by the algorithm is of the

1991 *Mathematics Subject Classification.* Primary 90C05; Secondary 65K05.

*Key words and phrases.* linear programming, face, least squares, LU factorization, degeneracy.

squares order, compared with the first order reduction by the simplex algorithm. The search direction is shown to be the best in certain sense. Some anti-degenerate tactics are incorporated by taking advantage of the face framework.

We will use $e_j$ to denote the unit vector with the $j$-th component 1, $I_j$ the $j \times j$ unit matrix, and $a_j$ the $j$-indexed column of matrix $A$, and so on. Without confusion, notations for matrices are also used for the corresponding index sets of their columns.

## 2. Outline of the Original Face Method

The search direction used is crucial for an LP solver. In this Section, we outline the original face method, focusing on its search direction.

Let $\bar{x}$ be a feasible solution to problem (1.1) and $A = (B, N)$ be a partition such that $\bar{x}_N = 0$. If $B$ is a $m \times k$ ($k \geq m$) matrix with full row rank $m$, $B$ and $N$ are termed *face matrix* and *inactive matrix*, respectively. The corresponding index sets of columns are termed similarly.

**Definition 2.1.** *The face, associated with $B$, is defined as*

$$P_B = \{x \mid Bx_B = b;\ x_B \geq 0,\ x_N = 0\}. \tag{2.1}$$

A face is *level face* if the objective value over it is constant. If the constant is equal to the optimal value, the face is called *optimal face*.

In each iteration, the face algorithm utilizes the so-called *face subprogram* blow:

$$\begin{array}{ll} \min & c_B^T x_B, \\ \text{s.t.} & Bx_B = b,\ x_B \geq 0, \end{array} \tag{2.2}$$

which amounts to minimizing the objective function of (1.1) over face $P_B$. Obviously, $\bar{x}_B$ is a feasible solution to (2.2).

It is known that the orthogonal projection onto the null of $B$ is

$$P = I - B^{\mathrm{T}}(BB^T)^{-1}B. \tag{2.3}$$

Hence, the projection of the objective gradient $c_B$ onto the null is

$$r = Pc_B = c_B - B^{\mathrm{T}}\hat{y}, \qquad \hat{y} = (BB^T)^{-1}Bc_B. \tag{2.4}$$

Vector $-r$ is a desirable search direction in the face, since it is the "steepest downhill" in the null of $B$, with respect to the objective.

To determine $\hat{y}$ and hence $-r$, both algorithms proposed in [9] and [11], respectively, handle the so-called *normal system*

$$BB^{\mathrm{T}}y = Bc_B. \tag{2.5}$$

The former initiates Cholesky factorization of $BB^T$ and updates the Cholesky factor thereafter, while the latter initiates and updates $(BB^T)^{-1}$ directly.

If $-r$ is nonzero, it is used to form a line search scheme to update the feasible solution $\bar{x}_B$. In the other case, as a level face is reached, optimality is tested. If optimality can not be declared, the face and inactive sets are adjusted, and the next iteration is carried out.

On the other hand, it is known that $\hat{y}$ is nothing but the unique solution to the following least squares problem:

$$\min \|c_B - B^T y\|_2. \tag{2.6}$$

and $r$ is just the corresponding residual. Thus, it is possible to derive a face algorithm by coping (2.6) instead. One idea along this line, for instance, is to premultiply both $c_B$ and $B^T$ by an orthogonal matrix $Q$ to convert problem (2.6) to its equivalent form below:

$$\min \|Qc_B - QB^T y\|_2.$$

As a result, if $Q$ is determined such that $QB^T$ is upper triangular, it is straightforward to obtain the wanted $(\hat{y}, r)$.

However, we will not go in that way, but deal with (2.6) by utilizing Gaussian elimination, alternatively.

## 3. Derivation of Search Direction

To begin with, partition face matrix $B$ into $(B_1, B_2)$, where $B_1 \in \mathcal{R}^{m\times m}$ with full rank $m$ is called *basis (matrix)*, and $B_2 \in \mathcal{R}^{m\times(k-m)}$ $(k \geq m)$ called *active matrix*. Associated items, such as indices, variables and so on, are said to be *basic* and *active*, respectively. Thereafter, if not indicated otherwise, we assume $k > m$, leaving out the trivial case $k = m$.

Then, the least squares problem (2.6) can be written

$$\min \left\| \begin{bmatrix} c_{B_1} \\ c_{B_2} \end{bmatrix} - \begin{bmatrix} B_1^T \\ B_2^T \end{bmatrix} y \right\|_2 \tag{3.1}$$

By a series of Gauss transformations, the coefficient matrix of the preceding can be converted to upper triangular. Assume that $L$ is the product of these transformations such that

$$LB^T = L \begin{bmatrix} B_1^T \\ B_2^T \end{bmatrix} = \begin{bmatrix} U_1 \\ 0 \end{bmatrix} \triangleq U, \tag{3.2}$$

where $U_1$ is non-singular upper triangular. Introduce notation

$$Lc_B = L \begin{bmatrix} c_{B_1} \\ c_{B_2} \end{bmatrix} = \begin{bmatrix} \tilde{c}_{B_1} \\ \tilde{c}_{B_2} \end{bmatrix} \triangleq \tilde{c}_B \tag{3.3}$$

Then, premultiplying both terms in (3.1) by $L$ leads to the following least squares problem:

$$\min \left\| \begin{bmatrix} \tilde{c}_{B_1} \\ \tilde{c}_{B_2} \end{bmatrix} - \begin{bmatrix} U_1 \\ 0 \end{bmatrix} y \right\|_2 \tag{3.4}$$

**Lemma 3.1.** *The least squares problem (3.4) has unique solution*

$$\tilde{y} = U_1^{-1}\tilde{c}_{B_1}, \tag{3.5}$$

*corresponding to the residual*

$$\tilde{z}_B = \begin{bmatrix} 0 \\ \tilde{c}_{B_2} \end{bmatrix}. \tag{3.6}$$

Proof. Consider the convex function

$$q(y) = \left\| \begin{bmatrix} \tilde{c}_{B_1} \\ \tilde{c}_{B_2} \end{bmatrix} - \begin{bmatrix} U_1 \\ 0 \end{bmatrix} y \right\|_2^2,$$

the gradient of which is

$$\nabla q(y) = \begin{bmatrix} U_1 \\ 0 \end{bmatrix}^T \begin{bmatrix} \tilde{c}_{B_1} \\ \tilde{c}_{B_2} \end{bmatrix} - \begin{bmatrix} U_1 \\ 0 \end{bmatrix}^T \begin{bmatrix} U_1 \\ 0 \end{bmatrix} y.$$

By using (3.5), it can be verified that $\nabla q(\tilde{y}) = 0$. Therefore, $\tilde{y}$ is the unique solution to (3.4). The corresponding residual is then

$$\begin{bmatrix} \tilde{c}_{B_1} \\ \tilde{c}_{B_2} \end{bmatrix} - \begin{bmatrix} U_1 \\ 0 \end{bmatrix} U_1^{-1} \tilde{c}_{B_1} = \begin{bmatrix} 0 \\ \tilde{c}_{B_2} \end{bmatrix},$$

which gives (3.6). □

Although (3.4) is not equivalent to (3.1) or (2.6), and hence the residual $\tilde{z}_B$ differs from the desired residual, $r$, of (2.6), it does help form the valuable vector below:

$$\Delta x_B = L^T \tilde{z}_B. \tag{3.7}$$

With this respect, the following result is claimed.

**Theorem 3.1.** *Assume that $\tilde{z}_B$ defined by (3.6) is nonzero. Then, $\Delta x_B$ determined by (3.7) satisfies the following conditions:*

$$B\Delta x_B = 0, \tag{3.8}$$

$$c_B^T \Delta x_B > 0. \tag{3.9}$$

Proof. Note that $\tilde{z}_B \neq 0$ implies $\tilde{c}_{B_2} \neq 0$ $(k > m)$.

From (3.2) and (3.6), it follows that

$$U^T \tilde{z}_B = \begin{bmatrix} U_1 \\ 0 \end{bmatrix}^T \begin{bmatrix} 0 \\ -\tilde{c}_{B_2} \end{bmatrix} = 0. \tag{3.10}$$

Then, (3.7) together with (3.2) and (3.10) leads to

$$B\Delta x_B = BL^T \tilde{z}_B = U^T \tilde{z}_B = 0,$$

which proves (3.8).

In addition, by (3.3), (3.6) and $\tilde{c}_{B_2} \neq 0$, it holds that

$$(\tilde{c}_B)^T \tilde{z}_B = \begin{bmatrix} \tilde{c}_{B_1} \\ \tilde{c}_{B_2} \end{bmatrix}^T \begin{bmatrix} 0 \\ \tilde{c}_{B_2} \end{bmatrix} = \tilde{c}_{B_2}^T \tilde{c}_{B_2} > 0,$$

which along with (3.3) and (3.7) gives

$$c_B^T \Delta x_B = c_B^T L^T \tilde{z}_B = (Lc_B)^T \tilde{z}_B = \tilde{c}_B^T \tilde{z}_B > 0.$$

Therefore, (3.9) is valid. □

The preceding Lemma ensures that vector $-\Delta x_B \neq 0$ is not only in the null of $B$ but also a downhill in the face defined by (2.1), and is hence eligible to be the search direction.

Formula (3.7) can be simplified if $L \in \mathcal{R}^{k\times k}$ is expressed in a partitioned form. In fact, it is known by matrix algebra that as the product of the Gauss transformations, $L$ is a unit lower triangular matrix, can be partitioned as

$$L = \begin{bmatrix} L_1 & 0 \\ H & I_{k-m} \end{bmatrix}, \tag{3.11}$$

where $L_1 \in R^{m\times m}$ is unit lower triangular.

**Theorem 3.2.** *Under the same assumption of Theorem 3.1, for $H$ defined by (3.11) and $\tilde{c}_{B_2}$ defined by (3.3), it holds that*

$$\Delta x_B = \begin{bmatrix} H^T \tilde{c}_{B_2} \\ \tilde{c}_{B_2} \end{bmatrix}. \tag{3.12}$$

Proof. From (3.7), (3.11) and (3.6), it follows that

$$\Delta x_B = L^T \tilde{z}_B = \begin{bmatrix} L_1^T & H^T \\ 0^T & I_{k-m} \end{bmatrix} \begin{bmatrix} 0 \\ \tilde{c}_{B_2} \end{bmatrix},$$

which leads to (3.12). □

Consequently, the determination of the search direction $-\Delta x_B$ comes down to computing $H$, a submatrix of $L$. Accumulating the Gauss transformations to gain $L$ seems not to be suitable for large sparse computing. Rather, it would be more practicable to maintain $L$ in a factored form. We will not do that, however, but take a more favorable approach, alternatively.

## 4. Practicable Search Direction

The trick is to leave subprogram (2.2) itself and switch to its equivalent simple form instead.

With partition $A = (B_1, B_2, N)$, problem (1.1) can be put into the following tableau:

(4.1)

| $x_{B_1}$ | $x_{B_2}$ | $x_N$ | $f$ | rhs |
|---|---|---|---|---|
| $B_1$ | $B_2$ | $N$ | | $b$ |
| $c_{B_1}^{\mathrm{T}}$ | $c_{B_2}^{\mathrm{T}}$ | $c_N^T$ | $-1$ | |

By the Gauss-Jordan elimination, the preceding can be converted to

(4.2)

| $x_{B_1}$ | $x_{B_2}$ | $x_N$ | $f$ | rhs |
|---|---|---|---|---|
| $I_m$ | $\bar{B}_2$ | $\bar{N}$ | | $\bar{b}$ |
| | $\bar{c}_{B_2}^T$ | $\bar{c}_N^T$ | $-1$ | $-\bar{f}$ |
| $\bar{x}_{B_1}^T$ | $\bar{x}_{B_2}^T$ | $*$ | - | - |

where current feasible solution $\bar{x}$ is inserted as the bottom row, with symbol "$*$" in place of $\bar{x}_N = 0$ marking out its inactive section. Tableau (4.2) is termed *canonical tableau* of subprogram (2.2).

Between the preceding and (4.1) (or (2.2)) , there is the following relationship:

$$\bar{B}_2 = B_1^{-1} B_2, \qquad \bar{b} = B_1^{-1} b, \tag{4.3}$$

$$\bar{c}_{B_2}^T = c_{B_2}^T - \bar{y}^T B_2, \qquad \bar{y}^T = c_{B_1}^T B_1^{-1}. \tag{4.4}$$

It is also implied that

$$\bar{B} = B_1^{-1} B = (\bar{B}_1, \bar{B}_2) \stackrel{\triangle}{=} (I_m, \bar{B}_2), \tag{4.5}$$

$$\bar{c}_B^T = c_B^T - \bar{y}^T B = (\bar{c}_{B_1}^T, \bar{c}_{B_2}^T) \stackrel{\triangle}{=} (0, \bar{c}_{B_2}^T). \tag{4.6}$$

The face section of tableau (4.2) represents the following *canonical* subprogram:

$$\begin{array}{ll} \min & \bar{c}_{B_2}^T x_{B_2}, \\ \text{s.t.} & x_{B_1} + \bar{B}_2 x_{B_2} = \bar{b},\ x_{B_1}, x_{B_2} \geq 0, \end{array} \tag{4.7}$$

Now, the derivation of the search direction made for (2.2) in the previous Section applies to (4.7), as carried over as follows.

The counterpart of the least squares problem (3.1) is then

$$\left\| \begin{bmatrix} 0 \\ \bar{c}_{B_2} \end{bmatrix} - \begin{bmatrix} I_m \\ \bar{B}_2^T \end{bmatrix} y \right\|_2. \tag{4.8}$$

as possible By premultiplying both terms by Gauss transformations, the lower submatrix $\bar{B}_2^T$ of the preceding coefficient matrix can be eliminated. Assume that $L'$ is the product of such Gauss transformations. By matrix algebra, $L'$ is of the partitioned form below:

$$L' = \begin{bmatrix} I_m & 0 \\ -\bar{B}_2^T & I_{k-m} \end{bmatrix}. \tag{4.9}$$

Thereby, the counterpart of expression (3.12) becomes fair simple, i.e.,

$$\Delta x_B = \begin{bmatrix} -\bar{B}_2 \bar{c}_{B_2} \\ \bar{c}_{B_2} \end{bmatrix}, \tag{4.10}$$

where it is noticeable that $\Delta x_{B_2}$ is equal to active reduced costs $\bar{c}_{B_2}$, which is available.

A a result, the counterparts of expressions (3.8) and (3.9) hold with respect to subprogram (4.7), i.e.,

$$\bar{B}\Delta x_B = 0, \tag{4.11}$$

$$\bar{c}_B^T \Delta x_B > 0. \tag{4.12}$$

Furthermore, vector $\Delta x_B$ satisfies similar expressions associated to subprogram (2.2) itself, as the following says.

**Theorem 4.1.** *If $\Delta x_B$ defined by (4.10) is nonzero, it satisfies*

$$B\Delta x_B = 0, \tag{4.13}$$

$$c_B^T \Delta x_B > 0. \tag{4.14}$$

Proof. From (4.5) and (4.11), it follows that

$$B\Delta x_B = B_1(B_1^{-1}B)\Delta x_B = B_1(\bar{B}\Delta x_B) = 0, \tag{4.15}$$

which gives (4.13).

On the other hand, it holds by (4.6) that

$$c_B^T = \bar{c}_B^T + \bar{y}^T B.$$

Postmultiplying both sides of the preceding together with (4.15) leads to

$$c_B^T \Delta x_B = \bar{c}_B^T \Delta x_B + \bar{y}^T B \Delta x_B = \bar{c}_B^T \Delta x_B, \tag{4.16}$$

which along with (4.12) proves (4.14). □

Theorem 4.1 ensures that $-\Delta x_B$ defined by (4.10) is a qualified search direction for (2.2), as well as for (4.7). Not only so, but it turns out that it is actually the "best" search direction in certain sense (see Section 10).

## 5. Updating Feasible Solution

Assume that $\bar{x}_B$ is current feasible solution to subprogram (2.2), and a nonzero search direction $\Delta x_B$ defined by (4.10) is available. Then, it is straightforward to form an update formula.

To begin with, we assume the following precondition:

$$S' \triangleq \{j \in B_2 \mid \bar{x}_j = 0\} = \emptyset, \qquad \text{or else} \qquad \bar{c}_j < 0, \ \forall\ j \in S', \tag{5.1}$$

which eliminates zero step-size risk caused from $B_2$ (see below).

Define index set

$$J = \{j \in B \mid \Delta x_j > 0\}. \tag{5.2}$$

In the case when $J$ is empty, we are then done, as the following says.

**Theorem 5.1.** *Assume that $\bar{x}_B$ is a feasible solution and $\Delta x_B$ defined by (4.10) is nonzero. If $J$ defined by (5.2) is empty, problem (1.1) is lower unbounded.*

Proof. Emptiness of $J$ implies $\Delta x_B \leq 0$, which along with $\bar{x}_B \geq 0$ leads to

$$\hat{x}_B = \bar{x}_B - \alpha \Delta x_B \geq 0, \qquad \forall\ \alpha \geq 0. \tag{5.3}$$

From the preceding, (4.13) and $B\bar{x}_B = b$, it follows that

$$B\hat{x}_B = B\bar{x}_B - \alpha B \Delta x_B = b, \qquad \forall\ \alpha \geq 0.$$

Therefore, $\hat{x}_B$ is a feasible solution to (2.2) for all $\alpha \geq 0$.

Moreover, by (5.3) and (4.14), it holds that

$$c_B^T \bar{x}_B - c_B^T \hat{x}_B = \alpha c_B^T \Delta x_B > 0, \qquad \forall\ \alpha > 0.$$

Consequently, the decrement in objective value goes to infinity, as so does $\alpha$. □

Assume now that $J$ is nonempty, and hence the following line search scheme is well-defined:

$$\hat{x}_B = \bar{x}_B - \alpha \Delta x_B, \tag{5.4}$$

where $\alpha$ is the step-size such that

$$\alpha = \min_{j \in J} \bar{x}_j / \Delta x_j \geq 0. \tag{5.5}$$

For the new solution $\hat{x}_B$, the following is claimed.

**Theorem 5.2.** *Assume that $\Delta x_B$ is defined by (4.10), and $J$ defined by (5.2) is nonempty. If $\bar{x}_B$ is a feasible solution to subprogram (2.2), then so is $\hat{x}_B$ given by (5.4).*

Proof. Under the assumption, it is clear that

$$B\bar{x}_B = b, \qquad \bar{x}_B \geq 0. \tag{5.6}$$

By (5.4), (4.13) and the first expression in (5.6), it holds that

$$B\hat{x}_B = B\bar{x}_B - \alpha B \Delta x_B = b. \tag{5.7}$$

Thus, $\hat{x}_B$ satisfies the equality constraints.

Note that $\bar{x}_j \geq 0\ \forall j \in B$. Without loss of generality, assume that $B \backslash J \neq \emptyset$.

It is clear by (5.2) that

$$\Delta x_j \leq 0, \quad j \in B \backslash J,$$

from which and (5.4), for any $\alpha \geq 0$, it follows that

$$\hat{x}_j = \bar{x}_j - \alpha \Delta x_j \geq 0, \qquad j \in B \backslash J. \tag{5.8}$$

On the other hand, it holds by (5.5) that

$$\bar{x}_j / \Delta x_j \geq \alpha \geq 0, \qquad j \in J.$$

Multiplying the preceding by $\Delta x_j > 0$ leads to

$$\bar{x}_j \geq \alpha \Delta x_j \geq 0, \qquad j \in J,$$

which together with (5.4) gives

$$\hat{x}_j = \bar{x}_j - \alpha \Delta x_j \geq 0, \qquad j \in J. \tag{5.9}$$

Consequently, (5.7), (5.8) and (5.9) together imply that $\hat{x}_B$ is a feasible solution to (4.7). □

From (5.8) and (5.9), it is seen that only the requirement $\hat{x}_j \geq 0, \forall\, j \in J$ limits the increase of the step-size, so that $\alpha$ determined by (5.5) is the largest possible step-size that can be taken along the search direction. Indices in $J$ and corresponding variables are often said to be *blocking* ones.

Note that $k$-vector $x_B$ is a feasible solution to (4.7) if and only if the corresponding $n$-vector

$$x = \begin{bmatrix} x_B \\ x_N \end{bmatrix} = \begin{bmatrix} x_B \\ 0 \end{bmatrix}$$

is a feasible solution to the original problem (1.1).

**Theorem 5.3.** *Under the same assumptions of Theorem 5.2, the objective value decreases by $\alpha\|\bar{c}_{B_2}\|_2^2$, with respect to problem (1.1) as well as with subprogram (4.7).*

Proof. From (4.6) and (4.10), it follows that

$$\bar{c}_B^T \Delta x_B = \begin{bmatrix} 0 \\ \bar{c}_{B_2} \end{bmatrix}^T \begin{bmatrix} \Delta x_{B_1} \\ \Delta x_{B_2} \end{bmatrix} = \bar{c}_{B_2}^T \bar{c}_{B_2}.$$

Then, (5.4), (4.16) and the preceding together lead to

$$\bar{c}_B^T \bar{x}_B - \bar{c}_B^T \hat{x}_B = \alpha \bar{c}_B^T \Delta x_B = \alpha \|\bar{c}_{B_2}\|_2^2.$$

So, the claim is valid for subprogram (4.7).

The claim is also valid for problem (1.1) since it is easily derived that

$$c^T \bar{x} - c^T \hat{x} = c_B^T \bar{x}_B - c_B^T \hat{x}_B = \bar{c}_B^T \bar{x}_B - \bar{c}_B^T \hat{x}_B = \alpha \|\bar{c}_{B_2}\|_2^2.$$

□

Theorem 5.3 indicates that given step-size $\alpha$, the decrement in objective value is proportional to the square of $\|\bar{c}_{B_2}\|_2$ (or $\|\Delta x_{B_2}\|_2$)! Surprisingly, this decrease is independent of $\Delta x_{B_1}$, but dependent on active reduced costs only. This feature is really remarkable, as one sees that the objective reduction by the simplex algorithm is only $-\alpha \bar{c}_q$ (where $q$ is the entering index). Nevertheless, if step-size $\alpha$ vanishes, so does the reduction. In fact, the so-called "new" feasible solution $\hat{x}_B$ yielded from (5.4) is then not new but the same as the old $\bar{x}_B$.

For step-size $\alpha$ given by (5.5), define nonempty subset of $J$ such that

$$J_\alpha = \{j \in J \mid \bar{x}_j / \Delta x_j = \alpha\}, \tag{5.10}$$

which is referred to as $\alpha$ *(level index) set*. Set $J_\alpha$ is closely related to zero step-size. If $\alpha = 0$, in particular, it is termed *null (level index) set*, denoted by $J_0$. Elements of it are termed *null indices*. There exists a null index if and only if $\alpha$ vanishes.

It is noticeable that under precondition (5.1), active set $B_2$ never includes any null indices, in other words, null set $J_0$ is entirely contained in basis $B_1$, if any. Therefore, if $\bar{x}_{B_1} > 0$, step-size $\alpha$ never vanishes.

**Definition 5.1.** *Feasible solution $\bar{x}_B$ is degenerate if $\bar{x}_{B_1}$ possesses a zero component. The associated index is said to be degenerate.*

Although degeneracy does not necessarily lead to zero step-size, it increases such risk. In fact, if a degenerate index belongs to $J$, it is actually a null index. We all know that for some real-world LP problems, in the simplex contest, there are large amounts of zero basic components arising, so that solution process gets stuck at some vertex for too long a time. Therefore, we try to incorporate possible anti-degenerate tactics by taking advantage of the face framework.

## 6. Anti-Degenerate Tactics

The basic idea behind our tactics is to try to put into the inactive set $N$ as many as possible indices, exhibiting complementarity-slackness.

Define subset $S$, consisting of all $B_2$'s complementarity-slackness indices, i.e.,

$$S = \{j \in B_2 \mid \bar{x}_j = 0,\ \bar{c}_j \geq 0\}. \tag{6.1}$$

Obviously, if $S$ is empty, precondition (5.1) is satisfied. In the other case, the precondition can be fulfilled by executing the following tactic.

**Tactic 6.1.** *If $S$ is nonempty, adjust $(B_2, N, k)$ by*

$$B_2 = B_2 \backslash S,\ N = N \cup S,\ k = k - |S|, \tag{6.2}$$

*where $|S|$ denotes the cardinality of $S$ (the same below).*

On the other hand, over a series of iterations, inactive set $N$ may include some non-complementarity-slackness indices, that is to say, the following index set is nonempty:

$$T = \{j \in N \mid \bar{c}_j < 0\}. \tag{6.3}$$

Therefore, it is needed to move all indices in $T$ to active set $B_2$ by enforce the following tactic:.

**Tactic 6.2.** *To expand the face, adjust $(B_2, N, k)$ by*

$$\hat{B}_2 = B_2 \cup T, \qquad \hat{N} = N \backslash T, \qquad k = k + |T| \tag{6.4}$$

In the rest of this Section, some additional anti-degenerate tactics are set forth in conjunction with a single contraction iteration, in which the number $k$ of columns of face matrix $B$ is strictly reduced.

It is assumed throughout that $J$ is nonempty, and new feasible solution $\hat{x}_B$ is computed by (5.4) with $\alpha$ determined by (5.5).

Let $J_\alpha$ be $\alpha$ set, defined by (5.10). It is clear that if $j \in J_\alpha$, $\hat{x}_j$ vanishes, and vice versa. In practice, it would be very often when $J_\alpha$ contains a large number of elements, resulting in zero steps following on. Thus, it is wise to move indices from $J_\alpha$ to inactive set $N$, as far as possible.

In particular, here is what happens with new $\hat{x}_{B_2}$.

**Proposition 6.1.** *Assume that $J_\alpha$ is defined by (5.10). If $\alpha > 0$, it holds that*

$$\hat{x}_j > 0, \qquad j \in B_2 \backslash J_\alpha. \tag{6.5}$$

Proof. By (4.10), (5.4) and Theorem 5.2, it holds that

$$\hat{x}_j = \bar{x}_j - \alpha \bar{c}_j \geq 0, \qquad j \in B_2 \backslash J_\alpha. \tag{6.6}$$

Then, the claim is justified in the following three cases:

Case (a): $\bar{c}_j < 0$. In this case, by $\bar{x}_j \geq 0$ and $\alpha > 0$, it is clear that $\hat{x}_j$ determined by (6.6) is strictly positive.

Case (b): $\bar{c}_j = 0$. In this case, combining $\hat{x}_j = 0$ and (6.6) leads to $\bar{x}_j = 0$. This contradicts precondition (5.1). Therefore, it holds that $\hat{x}_j > 0$.

Case (c): $\bar{c}_j > 0$. In this case, combining $\hat{x}_j = 0$ and (6.6) gives

$$\bar{x}_j / \bar{c}_j = \alpha,$$

which implies $j \in J_\alpha$, contradicting $j \in B_2 \backslash J_\alpha$. Therefore, $\hat{x}_j > 0$ is valid. □

There are the following two types of iterations.

(i) Simple iteration: $\tilde{J} \triangleq B_2 \cap J_\alpha \neq \emptyset$.

In this case, $B_2$ does contain all or part of $J_\alpha$'s elements, and the condition, $\alpha > 0$, of Proposition 6.1 can be dropped, as the following claims.

**Proposition 6.2.** *Assume that $J_\alpha$ is defined by (5.10). If $\tilde{J} \neq \emptyset$, impression (6.5) is valid.*

Proof. By Proposition 6.1, it is only needed to show $\alpha > 0$ in this case.

Assume on the contrary that $\alpha = 0$. Then, it follows from (5.10) that

$$\bar{x}_j/\bar{c}_j = 0, \qquad j \in \tilde{J},$$

which and $\bar{c}_j > 0$ ($j \in \tilde{J} \subset J$) together contradict precondition (5.1). Therefore, $\alpha > 0$, and hence the claim is valid. □

By the preceding Proposition, if $\tilde{J} \neq \emptyset$, moving $\tilde{J}$'s all elements from $B_2$ to $N$ fulfills precondition (5.1). In fact, for the new solution, $\tilde{J}$ contains all complementarity-slackness indices of $B_2$. Thus, we enforce the following tactic.

**Tactic 6.3.** *For simple iteration, adjust $(B_2, N, k)$ by*

$$\hat{B}_2 = B_2 \backslash \tilde{J},\ \hat{N} = N \cup \tilde{J},\ \hat{k} = k - |\tilde{J}|. \tag{6.7}$$

The associated computing effort is quite cheap, since there will be no need for basis change in this case. This is why such an iteration is said to be *simple*.

(ii) Full iteration: $\tilde{J} \triangleq B_2 \cap J_\alpha = \emptyset$. This is the case if and only if $J_\alpha \subset B_1$.

Now, there is a need for performing a pivot selection and basis change. To do so, designate by $j_i$ the index of the $i$-th column, $e_i$, of $\bar{B}_1 \triangleq I_m$. Note that the associated variable $x_{j_i}$ corresponds to the $i$-th row of tableau (4.2).

At first, select a pivot row by the following Rule.

**Rule 6.1.** *[Row Rule] Select row index $p$ such that*

$$j_p \in \arg\max_{j_i \in J_\alpha} \Delta x_{j_i}. \tag{6.8}$$

Thus, index $j_p$ is determined to leave $B_1$. The reason for choosing a such one is as follows.

It is evident that the cosine of the angle between the search direction $-\Delta x_B$ and the gradient of a constraint $x_j \geq 0$ equals

$$\alpha_j = -\Delta x_j/\|\Delta x_B\|, \qquad \forall j \in B. \tag{6.9}$$

By the so-called "heuristic characterization of optimal solution" (pp.56-60,[9]), for all $j_i \in B_1$, in particular, the constraint $x_{j_i} \geq 0$ corresponding to the smallest $\alpha_{j_i}$, or the largest $\Delta x_{j_i}$, tends to be active at an optimal solution. This means that the corresponding index should be selected to enter $N$, and the associated row should be taken as the pivot row.

Assume that $j_p \in B_1$ has been determined to enter $N$. The next thing to do is to select a pivot column.

To do so, bring up the $p$-th row of $\bar{B}_2$ from canonical face tableau (4.2), i.e.,

$$v_{B_2}^T = e_p^T \bar{B}_2. \tag{6.10}$$

The following claim can be made.

**Proposition 6.3.** *There exists index $j \in B_2$ such that $v_j \bar{c}_j < 0$.*

Proof. Assume, on the contrary, that

$$v_j \bar{c}_j \geq 0, \qquad \forall j \in B_2,$$

from which it follows that

$$v_{B_2}^T \bar{c}_{B_2} = \sum_{j \in B_2} v_j \bar{c}_j \geq 0. \tag{6.11}$$

On the other hand, since $j_p \in J_\alpha \subset J$, it is evident that

$$\Delta x_{j_p} > 0. \tag{6.12}$$

In addition, by (6.10) and (4.5), it holds that

$$e_p^T \bar{B} = e_p^T (I_m, \bar{B}_2) = (e_p^T, v_{B_2}^T),$$

which and (4.11) together gives

$$0 = e_p^T \bar{B} \Delta x_B = (e_p^T, v_{B_2}^T) \begin{bmatrix} \Delta x_{B_1} \\ \Delta x_{B_2} \end{bmatrix} = \Delta x_{j_p} + v_{B_2}^T \Delta x_{B_2}.$$

The preceding along with (4.10) and (6.12) leads to

$$v_{B_2}^T \bar{c}_{B_2} = -\Delta x_{j_p} < 0, \tag{6.13}$$

which contradicts (6.11). Therefore, the statement is valid. □

In view of $\Delta x_{B_2} = \bar{c}_{B_2}$, we have the following plausible preferences for bringing from $B_2$ to enter $B_1$ such an index $q$ that corresponds to:

(1) the minimal $\bar{c}_j$;
   According to the heuristic characterization, it is reasonable to select from $B_2$ an entering index, corresponding to the maximal $\alpha_j$ defined by (6.9), or minimal $\bar{c}_j$.
(2) the maximal $\bar{x}_j$;
   Such a choice increases possibility of entering a non-degenerate index.
(3) the minimal $v_j \bar{c}_j < 0$;
   According to Proposition 6.3, the minimal is negative.
   In addition, it holds by (6.13) that

$$\Delta x_{j_p} = -v_{B_2}^T \bar{c}_{B_2} = -\sum_{j \in B_2} v_j \bar{c}_j > 0.$$

   Therefore, such a choice corresponds to the maximal term in $\Delta x_{j_p}$.
(4) the minimal $\bar{c}_j / vj < 0$ ($v_j \neq 0$);
   For the same reason of the preceding item, the minimal is negative.
   It is seen that the current value zero of the leaving $\bar{c}_{j_p}$ will become $-\bar{c}_j / v_j$, which tends to be a large positive number, after the related basis change. Therefore, such a choice reduces the likelihood that leaving index $j_p$ re-enters $B_1$, as it tends to stay in $N$.
(5) the maximal $|v_j|$.
   This choice is conducive to numerical stability.

Thus, there exist multiple options for column pivoting. We only bring up the following one using sets

$$\begin{aligned} J_1 &= \arg\min\{\bar{c}_j \mid v_j \neq 0,\ j \in B_2\} \\ J_2 &= \arg\max_{j\in J_1} \bar{x}_j \end{aligned}$$

**Rule 6.2.** *[Column Rule] For entering $B_1$, select an index $q \in J_1$; in case of a tie, select $q$ from $J_2$; if there is still a tie, determine*

$$q \in \arg\max_{j\in J_2} |v_j|. \tag{6.14}$$

Assume now that row index $p$ and column index $q$ have been determined. We update the canonical face tableau (4.2), just as in the simplex context, and then adjust sets accordingly by the following tactic.

**Tactic 6.4.** *For full iteration, adjust $(B_1, B_2, N, k)$ by*

$$\hat{B}_1 = (B_1\backslash\{j_p\}) \cup \{q\},\ \hat{B}_2 = B_2\backslash\{q\},\ \hat{N} = N \cup \{j_p\},\ k = k-1. \tag{6.15}$$

It is clear that Proposition 6.1 applies to case of $\tilde{J} = \emptyset$. Therefore, if $\alpha > 0$, executing Tactic 6.4 leads to $\hat{x}_q > 0$ and $\hat{x}_{B_2} > 0$, fulfilling precondition (5.1). If $\alpha = 0$, however, there is no such a favorable outcome.

Face contraction process performs a series of contraction iterations, corresponding to subprograms of form (4.7) with decreasing $k$, until reaching a level face to test for optimality or detecting lower unboundedness.

## 7. Optimality Test

It is noticeable that (4.2) is actually a simplex tableau of the original problem (1.1) if one takes into account of both the face and inactive sections.

Assume that search direction $\Delta x_B$ vanishes, and hence line search scheme (5.4) becomes meaningless. It is clear that this case occurs if and only if $k = m$, or else $\bar{c}_{B_2} = 0$.

**Lemma 7.1.** *If active reduced costs $\bar{c}_{B_2}$ vanishes, the face $P_B$ defined by (2.1) is a level face*

Proof. By $\bar{c}_{B_2} = 0$, it is clear that the equation corresponding to the objective row of (4.2) becomes

$$0x_{B_1} + 0x_{B_2} + \bar{c}_N^T x_N - f = -\bar{f}. \tag{7.1}$$

Since all $\bar{x} \in P_B$ satisfy $\bar{x}_N = 0$, substituting them to (7.1) results in $f = \bar{f}$, which means that $P_B$ is a level face. □

Note that face $P_B$ contains current feasible solution $\bar{x}$, listed in the bottom row of tableau (4.2).

The following says that optimality is achieved if, in addition to $\Delta x_B = 0$, it holds that $\bar{c}_N \geq 0$.

**Theorem 7.1.** *Assume that active reduced costs $\bar{c}_{B_2}$ vanishes. Then $P_B$ is an optimal face if set $T$ defined by (6.3) is empty:*

Proof. Note that emptiness of $T$ implies $\bar{c}_N \geq 0$.

By Lemma 7.1, $\bar{c}_{B_2} = 0$ implies that $P_B$ is a level face. Since $\bar{x}$ belongs to $P_B$, therefore, it is only needed to examine whether $\bar{x}$ is optimal. In fact, it is so since

$\bar{c}_{B_2} = 0$ and $\bar{c}_N \geq 0$ together ensure that the simplex tableau (4.2) is optimal. Therefore, $P_B$ is an optimal face. □

**Corollary 7.1.** *Under assumptions of Theorem 7.1, the associated $\bar{x}$ and $\bar{c}$ are a pair of primal and dual optimal solutions to problem (1.1).*

In case when $\bar{c}_{B_2}$ vanishes but $T$ is nonempty, and hence optimality cannot be claimed, one enforces tactic 6.2, fulfilling precondition (5.1) and leaving all complementarity-slackness indices to form a new inactive set $\hat{N}$. Consequently, he can get another contraction process started with an expanded face.

## 8. New Face Algorithm: Tableau Form

To begin with, assume that an arbitrary feasible solution $\bar{x}$ to (1.1) is available, and a basis $B_1$ is determined. Setting $B_2 = A\backslash B_1,\ N = \emptyset$, we generate an initial canonical face tableau of form (4.2) with feasible solution $\bar{x}_B$.

Although any set of $m$ independent columns of $A$ are eligible for being an initial basis $B_1$, in principle, it seems to be desirable to have one such that $\bar{x}_{B_1} > 0$ or nearly so. Indeed, it seems to be favorable to get started from a basis along with the related basic feasible (preferably non-degenerate) solution. In general, this helps set $S$ contain more complementarity slackness elements, so that moving $S$ from $B_2$ to inactive set $N$ results in a smaller face set $B$.

The face is then contracted iteration by iteration: whenever $\bar{c}_{B_2}$ is nonzero, $\bar{x}_B$ is updated with $(B_1, B_2, N, k)$ adjusted accordingly, until reaching a level face or detecting lower unboundedness. In case when optimality test fails, the face is expanded, and another contraction process is carried out.

The overall steps are summarized as follows.

*Algorithm* 8.1.[New face algorithm: Tableau form] Initial : Partition $(B_1, B_2 = A\backslash B_1, N = \emptyset),\ k = n$; canonical face tableau of form (4.2); feasible solution $\bar{x}$. This algorithm solves problem (1.1).

1. If $S$ defined by (6.1) is nonempty, adjust $(B_2, N, k)$ by (6.2).
2. Go to step 13 if $\bar{c}_{B_2} = 0$.
3. Compute $\Delta x_B$ by (4.10).
4. Stop if $J = \{j \in B \mid \Delta x_j > 0\}$ is empty.
5. Determine $\alpha$ by (5.5) and $J_\alpha$ by (5.10).
6. If $\alpha > 0$, update $\bar{x}_B$ by (5.4).
7. If $B_2 \cap J_\alpha \neq \emptyset$, adjust $(B_2, N, k)$ by (6.7);
   then if $k = m$, go to step 13, else to step 3.
8. Determine row index $p$ by (6.8).
9. Determine column index $q$ by (6.14).
10. Multiply the $p$-th row by $1/\bar{a}_{p,q}$, and add $-\bar{a}_{i,q}$ times of the $p$-th row to the $i$-th row for all $i = 1, \ldots, m+1;\ i \neq p$.
11. Adjust $(B_1, B_2, N, k)$ by (6.15).
12. If $k > m$, then if $\alpha > 0$, go to step 3, else to step 1.
13. Stop if $T$ defined by (6.3) is empty.
14. Adjust $(B_2, N, k)$ by (6.4).
15. Go to step 3.

Steps 1 through 15 constitute a full iteration, while steps 1 through 7 constitute a simple iteration.

**Theorem 8.1.** *Assume non-degeneracy in a single iteration for each contraction process. Algorithm 8.1 terminates either at*

*(i) step 4, declaring lower unboundedness; or at*

*(ii) step 13, giving an optimal face together with a pair of primal and dual optimal solutions.*

Proof. It is evident that Algorithm 8.1 reaches a level face at the end of every contraction process, within finitely many iterations, unless detecting lower unboundedness.

Assume that Algorithm 8.1 does not terminate. Then, it must carry out infinitely many contraction processes, each of which includes an iteration with nonzero step-size. Therefore, by Theorem 5.3, the traversed level faces correspond to strictly reduced objective values. This contradicts that the number of level faces is finite. Therefore, the algorithm terminates.

By Theorem 5.1, exit step 4 indicates lower unboundedness. By Theorem 7.1 and Corollary 7.1, exiting from step 13 gives an optimal face together with a pair of primal and dual solutions. □

We point out that degeneracy remains a problem, at this stage. But, the non-degeneracy assumption in Algorithm 8.1 seems to be quite relaxed. Indeed, it is unimaginable that a contraction process does not include any iteration involving nonzero step-size.

It is more than that. The leaving index, which is moved from $B_1$ to $N$ by Tactic 6.4, will never re-enter $B_1$ before another face contraction process carried out, if any. Moreover, if step-size is nonzero, both Tactics 6.3 and 6.4 ensue that all components of the new $\bar{x}_{B_2}$ are strictly positive. Consequently, solution's component corresponding to any would-be entering index never vanishes, lowering degeneracy risk.

It is well-known that the conventional simplex algorithm fails to solve Beale's simple problem because of cycling [1]. It seems to be interesting to see what will happen if the new face method is applied to the same problem.

**Example 8.1.** *Solve Beale's problem by Algorithm 8.1:*

$$\begin{array}{lllllllllll}
\min & f = -3/4x_4 + 20x_5 - 1/2x_6 + 6x_7, & & & & & & & & & \\
\text{s.t.} & x_1 & & & +1/4x_4 & -8x_5 & & -x_6 & +9x_7 & = & 0, \\
& & x_2 & & +1/2x_4 & -12x_5 & & -1/2x_6 & +3x_7 & = & 0, \\
& & & x_3 & & & & +x_6 & & = & 1, \\
& x_j \geq 0, & & j = 1, \cdots, 7. & & & & & & &
\end{array}$$

Initial: $\bar{x} = [0, 0, 1, 0, 0, 0, 0]^T$, $B_1 = \{1, 2, 3\}$, $B_2 = \{4, 5, 6, 7\}$, $N = \emptyset$.
Face tableau

| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ |
|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1/4 | −8 | −1 | 9 |
| 0 | 1 | 0 | 1/2 | −12 | −1/2 | 3 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | −3/4 | 20 | −1/2 | 6 |
| 0 | 0 | 1 | 0 | ∗ | 0 | ∗ |

Iteration 1:

1. $S = \{5, 7\} \neq \emptyset,\ B_2 = \{4, 6\},\ N = \{5, 7\}$; $\bar{x}_B = [1, 2, 3, 4, 6]^T,\ B = (B_1, B_2),\ k = 5.$
2. $\bar{c}_{B_2} = [-3/4, -1/2]^T \neq 0.$
3. $B_2 = \begin{bmatrix} 1/4 & -1 \\ 1/2 & -1/2 \\ 0 & 1 \end{bmatrix}$, $\Delta x_B = [-5/16, 1/8, 1/2, -3/4, -1/2]^T \neq 0.$
4. $J = \{2, 3\} \neq \emptyset.$
5. $\alpha = \min\{0, 2\} = 0,\ J_\alpha = \{2\}.$
7. $J_\alpha \subset B_1.$
8. $j_p = 2,\ p = 2.$
9. $\min\{-3/4, -1/2\} = -3/4,\ J_1 = \{4\}.$
10. Multiply row 2 by $1/(1/2)$, and add $-1/4, 3/4$ times of row 2 to rows 1,4, respectively:

| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ |
|---|---|---|---|---|---|---|
| 1 | $-1/2$ | 0 | 0 | $-2$ | $-3/4$ | $15/2$ |
| 0 | 2 | 0 | 1 | $-24$ | $-1$ | 6 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | $3/2$ | 0 | 0 | 2 | $-5/4$ | $21/2$ |
| 0 | $*$ | 1 | 0 | $*$ | 0 | $*$ |

11. $B_1 = \{1, 4, 3\}, B_2 = \{6\}, N = \{2, 5, 7\}, k = 4.$
12. $k > m = 4,\ \alpha = 0.$

Iteration 2:

1. $S = \emptyset.$
2. $\bar{c}_{B_2} = [-3/4, -1/2]^T \neq 0.$
3. $B_2 = [-3/4, -1, 1]^T$, $\Delta x_B = [-15/16, -5/4, 5/4, -5/4]^T \neq 0.$
4. $J = \{3\} \neq \emptyset.$
5. $\alpha = \min\{4/5\},\ J_\alpha = \{3\}.$
6. $\bar{x}_B = [0, 0, 1, 0]^T - (4/5)[-15/16, -5/4, 5/4, -5/4]^T = [3/4, 1, 0, 1]^T$
7. $J_\alpha \subset B_1.$
8. $j_p = 3,\ p = 3.$
9. $\min\{-5/4\},\ J_1 = \{6\}.$
10. Add $3/4, 1, 5/4$ times of row 3 to rows 1,2,4, respectively:

| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ |
|---|---|---|---|---|---|---|
| 1 | $-1/2$ | $3/4$ | 0 | $-2$ | 0 | $15/2$ |
| 0 | 2 | 1 | 1 | $-24$ | 0 | 6 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | $3/2$ | $5/4$ | 0 | 2 | 0 | $21/2$ |
| $3/4$ | $*$ | $*$ | 1 | $*$ | 1 | $*$ |

11. $B_1 = \{1, 4, 6\}, B_2 = \emptyset, N = \{2, 3, 5, 7\}, k = 3.$

13. $T = \emptyset$. Terminated with primal and dual optimal solutions, and optimal value:

$$x^* = [3/4, 0, 0, 1, 0, 1, 0]^T, \qquad z^* = [0, 3/2, 5/4, 0, 2, 0, 21/2]^T, \qquad f^* = -5/4.$$

Surprisingly enough, this cycling problem was solved within only two iterations!

## 9. New Face Algorithm

Algorithm 8.1 can be transformed into a practicable version, using original data instead of the canonical face tableau as a whole. It involves some key quantities, which can be calculated based on (4.3) and (4.4).

We favor the LU factorization. Assume that $B_1 = LU$ is available.

(i) Search direction $\Delta x_B$ can be calculated as follows.

At first, solve the following two $m \times m$ triangular systems successively for $\bar{y} = B_1^{-T} c_{B_1}$:

$$U^T w = c_{B_1}, \qquad L^T y = w. \tag{9.1}$$

Then compute active reduced costs by

$$\bar{c}_{B_2} = c_{B_2} - B_2^T \bar{y}. \tag{9.2}$$

Thereby, $\Delta x_{B_2} = \bar{c}_{B_2}$ is available.

To gain $\Delta x_{B_1} = -B_1^{-1} B_2 \bar{c}_{B_2}$, solve the following two $m \times m$ triangular systems successively:

$$Lw = -B_2 \bar{c}_{B_2}, \qquad U \Delta x_{B_1} = w. \tag{9.3}$$

(ii) Assume that pivot row index $p$ is determined. Then the corresponding row of $\bar{B}_2$, i.e., $v_{B_2}^T = e_p^T B_1^{-1} B_2$, can be obtained as follows:

Firstly, solve the following two $m \times m$ triangular systems successively for $u = B_1^{-T} e_p$:

$$U^T w = e_p, \qquad L^T u = w. \tag{9.4}$$

Then, compute

$$v_{B_2} = B_2^T u. \tag{9.5}$$

(iii) Let $v_{B_2}$ be available. Assume that $j_p \in B_1$ was moved to $N$ and $q \in B_2$ moved to $B_1$.

Then, active reduced costs can be updated for the next iteration by the following formula:

$$\hat{c}_{B_2} = \bar{c}_{B_2} + \beta v_{B_2}, \qquad \beta = -\bar{c}_q / v_q. \tag{9.6}$$

The overall steps are summarized in the following algorithm.

*Algorithm* 9.1.[New face algorithm] Initial : Given initial partition $(B_1, B2 = A \backslash B_1, N = \emptyset)$, $k = n$; factorization $B_1 = LU$; feasible solution $\bar{x}$. This algorithm solves problem (1.1).

1. Compute $\bar{c}_{B_2}$ by (9.1) and (9.2).
2. If $S$ defined by (6.1) is nonempty, adjust $(B_2, N, k)$ by (6.2).
3. Go to step 16 if $\bar{c}_{B_2} = 0$.
4. Compute $\Delta x_{B_1}$ by (9.3) and set $\Delta x_{B_2} = \bar{c}_{B_2}$.
5. Stop if $J = \{j \in B \mid \Delta x_j > 0\}$ is empty. (lower unbounded).
6. Determine $\alpha$ by (5.5), and $J_\alpha$ by (5.10).
7. If $\alpha > 0$, update $\bar{x}_B$ by (5.4).
8. If $B_2 \cap J_\alpha \neq \emptyset$, adjust $(B_2, N, k)$ by (6.7);

then if $k = m$, go to step 16, else to step 4.
9. Determine row index $p$ by (6.8).
10. Compute $v_{B_2}$ by (9.4) and (9.5).
11. Determine column index $q$ by (6.14).
12. Adjust $(B_1, B_2, N, k)$ by (6.15).
13. Update $L$ and $U$.
14. Update $\bar{c}_{B_2}$ by (9.6).
15. If $k > m$, then if $\alpha > 0$, go to step 4, else to step 1.
16. Compute $\bar{c}_N = c_N - N^T\bar{y}$, where $\bar{y}$ is yielded by (9.1).
17. Stop if $T$ defined by (6.3) is empty (optimality achieved).
18. Adjust $(B_2, N, k)$ by (6.4).
19. Go to step 4.

Note: Steps 1 through 15 constitute a full iteration, while steps 1 through 8 constitute a simple iteration.

**Example 9.1.** *Solve the following problem by Algorithm 9.1 in two phases:*

$$\begin{array}{lrrrrrrrcr}
\min & f = -6x_1 + 5x_2 - 3x_3 & -4x_5 + 9x_6 - 2x_7, & & & & & & & \\
\text{s.t.} & 3x_1 & -2x_2 & +5x_3 & +4x_4 & & +3x_6 & +5x_7 & = & 15, \\
 & & -x_2 & +6x_3 & +2x_4 & +8x_5 & -5x_6 & 4x_7 & = & 18, \\
 & 5x_1 & +3x_2 & -2x_3 & -8x_4 & -4x_5 & x_6 & -3x_7 & = & 9, \\
 & x_j \geq 0, & j = 1, \cdots, 7. & & & & & & &
\end{array}$$

Any conventional Phase-1 approach is applicable to provide a basis and a feasible solution to get Algorithm 9.1 started. Here, we form the auxiliary problem using artificial variables.

Phase-I:

Introduce artificial variables $x_8, x_9, x_{10}$ to the three equalities respectively, and eliminate them from auxiliary objective function $f' = -x_8 - x_9 - x_{10}$ by using the equalities. The resulting Phase-I problem is as follows:

$$\begin{array}{lrrrrrrrrrcr}
\min & f' = -8x_1 - 9x_3 + 2x_4 - 4x_5 + 7x_6 - 6x_7, & & & & & & & & & & \\
\text{s.t.} & 3x_1 & -2x_2 & +5x_3 & +4x_4 & & +3x_6 & +5x_7 & +x_8 & & = & 15, \\
 & & -x_2 & +6x_3 & +2x_4 & +8x_5 & -5x_6 & 4x_7 & & +x_9 & = & 18, \\
 & 5x_1 & +3x_2 & -2x_3 & -8x_4 & -4x_5 & x_6 & -3x_7 & & +x_{10} & = & 9, \\
 & x_j \geq 0, & j = 1, \cdots, 10. & & & & & & & & &
\end{array}$$

The objective gradient and the feasible solution are, respectively,

$$c = [-8, 0, -9, 2, -4, 7, -6, 0, 0, 0]^T \qquad \text{and} \qquad \bar{x} = [0, 0, 0, 0, 0, 0, 0, 15, 18, 9]^T.$$

Initially, we take

$$B_1 = \{8, 9, 10\},\ B_2 = \{1, 2, 3, 4, 5, 6, 7\},\ N = \emptyset,$$

so that $B_1$ and its LU factors are all unit matrices, and $\bar{x}_{B_2} = 0$.

Iteration 1:

1. $\bar{y} = 0$, $\bar{c}_{B_2} = c_{B_2} = [-8, 0, -9, 2, -4, 7, -6]^T$.
2. $S = \{2, 4, 6\} \neq \emptyset$, $B_2 = \{1, 3, 5, 7\}, N = \{2, 4, 6\}, k = 7$.

3. $\bar{c}_{B_2} \neq 0$.

4. $L = U = I_3,\ w = -B_2\bar{c}_{B_2} = -\begin{bmatrix} 3 & 5 & 0 & 5 \\ 0 & 6 & 8 & 4 \\ 5 & -2 & -4 & -3 \end{bmatrix}[-8,-9,-4,-6]^T$

$= [99, 110, -12]^T,\ \Delta x_B = [99, 110, -12, -8, -9, -4, -6]^T \neq 0.$

5. $J = \{8, 9\} \neq \emptyset$.
6. $\alpha = \min\{5/33, 9/55\} = 5/33,\ J_\alpha = \{8\}$.
7. $\bar{x}_B = [15, 18, 9, 0, 0, 0, 0]^T - (5/33)[99, 110, -12, -8, -9, -4, -6]^T$
$= [0, 4/3, 119/11, 40/33, 15/11, 20/33, 10/11]^T$.
8. $J_\alpha \in B_1$.
9. $j_p = 8,\ p = 1$.
10. $u = w = e_1,\ v_{B_2} = [3, 5, 0, 5]^T$
11. $\min\{-8, -9, -6\} = -9, J_1 = \{3\}, q = 3$.
12. $B_1 = \{3, 9, 10\}, B_2 = \{1, 5, 7\}, N = \{2, 4, 6, 8\}, k = 6$.

13. Permuted $L = \begin{bmatrix} 5/6 & 1 & 0 \\ 1 & 0 & 0 \\ -1/3 & -2/5 & 1 \end{bmatrix}, U = \begin{bmatrix} 6 & 1 & 0 \\ 0 & -5/6 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

14. $\bar{c}_{B_2} = [-8, -4, -6]^T + (-(-9)/5)[3, 0, 5]^T = [-13/5, -4, 3]^T$.
15. $k = 6 > m,\ \alpha > 0$.

Iteration 2:

4. $-B_2\bar{c}_{B_2} = -\begin{bmatrix} 3 & 0 & 5 \\ 0 & 8 & 4 \\ 5 & -4 & -3 \end{bmatrix}\begin{bmatrix} -13/5 \\ -4 \\ 3 \end{bmatrix} = \begin{bmatrix} -36/5 \\ 20 \\ 6 \end{bmatrix},\ w = \begin{bmatrix} 20 \\ -358/15 \\ 78/25 \end{bmatrix}$

$\Delta x_B = [-36/25, 716/25, 78/25, -13/5, -4, 3]^T \neq 0$.
5. $J = \{9, 10, 7\} \neq \emptyset$.
6. $\alpha = \min\{25/537, 1328/383, 10/33\} = 25/537,\ J_\alpha = \{9\}$.
7. $\bar{x}_B = [15/11, 4/3, 119/11, 40/33, 20/33, 10/11]^T - (25/537)[-36/25, 716/25, 78/25, -13/5, -4, 3]^T$
$= [1166/815, 0, 2839/266, 2625/1969, 1560/1969, 911/1184]^T$.
8. $J_\alpha \in B_1$.
9. $j_p = 9,\ p = 2$.
10. $e_2 = [0, 1, 0]^T,\ w = [0, -6/5, 0]^T,\ u = [-6/5, 1, 0]^T,\ v_{B_2} = [-18/5, 8, -2]^T$.
11. $\min\{-13/5, -4, 3\} = -4, J_1 = \{5\}, q = 5$.
12. $B_1 = \{3, 5, 10\}, B_2 = \{1, 7\}, N = \{2, 4, 6, 8, 9\}, k = 5$.

13. Permuted $L = \begin{bmatrix} 5/6 & 1 & 0 \\ 1 & 0 & 0 \\ -1/3 & 1/5 & 1 \end{bmatrix}, U = \begin{bmatrix} 6 & 8 & 0 \\ 0 & -20/3 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

14. $\bar{c}_{B_2} = [-13/5, 3]^T + (-(-4)/8)[-18/5, -2]^T = [-22/5, 2]^T$.
15. $k = 5 > m,\ \alpha > 0$.

Iteration 3:

4. $-B_2\bar{c}_{B_2} = -\begin{bmatrix} 3 & 5 \\ 0 & 4 \\ 5 & -3 \end{bmatrix}\begin{bmatrix} -22/5 \\ 2 \end{bmatrix} = \begin{bmatrix} 16/5 \\ -8 \\ 28 \end{bmatrix}$, $w = \begin{bmatrix} -8 \\ 148/15 \\ 584/25 \end{bmatrix}$,

$\Delta x_B = [16/25, -37/25, 584/25, -22/5, 2]^T \neq 0$.
5. $J = \{3, 10, 7\} \neq \emptyset$.
6. $\alpha = \min\{959/429, 514/1125, 911/2368\} = 911/2368,\ J_\alpha = \{7\}$.
7. $\bar{x}_B = [1166/815, 1560/1969, 2839/266, 2625/1969, 911/1184]^T$
$-(911/2368)[16/25, -37/25, 584/25, -22/5, 2]^T$
$= [1509/1274, 625/459, 3249/1927, 2687/888, 0]^T$.
8. $J_\alpha \in B_2, B_1 = \{3, 5, 10\}; B2 = \{1\},\ N = \{2, 4, 6, 7, 8, 9\}, k = 4 > m$.

Iteration 4:

4. $-B_2\bar{c}_{B_2} = -[3, 0, 5]^T[-22/5] = [66/5, 0, 22]^T$, $w = [0, 66/5, 484/25]$.
$\Delta x_B = [66/25, -99/50, 484/25, -22/5]^T \neq 0$.
5. $J = \{3, 10\} \neq \emptyset$.
6. $\alpha = \min\{485/1081, 143/1642\} = 143/1642,\ J_\alpha = \{10\}$.
7. $\bar{x}_B = [1509/1274, 625/459, 3249/1927, 2687/888, ]^T - (143/1642)[66/25, -99/50, 484/25, -22/5]^T$
$= [24191/25343, 135/88, 0, 75/22]^T$.
8. $J_\alpha \in B_1$.
9. $j_p = 10,\ p = 3$.
10. $e_3 = [0, 0, 1]^T$, $w = e_3$, $u = [-1/5, 1/2, 1]^T$, $v = [22/5]$.
11. $\min\{-22/5\}, J_1 = \{1\}, q = 1$.
12. $B_1 = \{3, 5, 1\}, B_2 = \emptyset, N = \{2, 4, 6, 7, 8, 9, 10\}, k = 3$.

13. Permuted $L = \begin{bmatrix} 5/6 & 1 & 0 \\ 1 & 0 & 0 \\ -1/3 & 1/5 & 1 \end{bmatrix}$, $U = \begin{bmatrix} 6 & 8 & 0 \\ 0 & -20/3 & 3 \\ 0 & 0 & 22/5 \end{bmatrix}$.

14. No need for updating $\bar{c}_{B_2}$.
15. $k = m = 3$.

With 4 iterations, including 3 full and 1 simple, Phase-I ended when all the artificial variables left the basis.

Phase-II:

The preceding Phase-1 produces an initial feasible solution

$$\bar{x} = [75/22, 0, 24191/25343, 0, 135/88, 0, 0]^T,$$

and the related basis $B_1 = \{3, 5, 1\}$ with (permuted) LU factors.

Referring to the original objective gradient, i.e.,

$$c = [-6, 5, -3, 0, -4, 9, -2]^T$$

and setting $B_2 = A \backslash B_1 = \{2, 4, 6, 7\}, N = \emptyset$, we get Algorithm 9.1 started for Phase-II as follows:

Iteration 5:

$c_B = [-3, -4, -6, 5, 0, 9, -2]^T,\ \bar{x}_B = [24191/25343, 135/88, 75/22, 0, 0, 0, 0]$.
1. $c_{B_1} = [-3, -4, -6]^T, w = [-1/2, 0, -15/11]^T$, $\bar{y} = [3/11, -13/11, -15/11]^T$;

$$\bar{c}_{B_2} = \begin{bmatrix} 5 \\ 0 \\ 9 \\ -2 \end{bmatrix} - \begin{bmatrix} -2 & 4 & -3 & 5 \\ -1 & 2 & -5 & 4 \\ 3 & -8 & 1 & -3 \end{bmatrix}^T \begin{bmatrix} 3/11 \\ -13/11 \\ -15/11 \end{bmatrix} = \begin{bmatrix} 93/11 \\ -106/11 \\ 58/11 \\ -30/11 \end{bmatrix}.$$

2. $S = \{2, 6\} \neq \emptyset$, $B_2 = \{4, 7\}$, $N = \{2, 6\}$.
3. $\bar{c}_{B_2} \neq 0$.

$$4.\ -B_2\bar{c}_{B_2} = -\begin{bmatrix} 4 & 5 \\ 2 & 4 \\ -8 & -3 \end{bmatrix} \begin{bmatrix} -106/11 \\ -30/11 \end{bmatrix} = \begin{bmatrix} 574/11 \\ 332/11 \\ -938/11 \end{bmatrix},\ w = \begin{bmatrix} 332/11 \\ 892/33 \\ -4434/55 \end{bmatrix}.$$

$\Delta x_B = [2593/121, -3038/247, -2217/121, -106/11, -30/11]^T \neq 0$.
5. $J = \{3\} \neq \emptyset$.
6. $\alpha = \min\{191/4288\}$, $J_\alpha = \{3\}$.
7. $\bar{x}_B = [24191/25343, 135/88, 75/22, 0, 0]^T - (191/4288)$
$[2593/121, -3038/247, -2217/121, -106/11, -30/11]^T$
$= [0, 2007/964, 7204/1705, 743/1731, 233/1918]^T$.
8. $J_\alpha \in B_1$.
9. $j_p = 3,\ p = 1$.
10. $w = [1/6, 1/5, -3/22]^T$, $u = [5/22, -3/44, -3/22]^T$, $v_{B_2} = [41/22, 14/11]^T$.
11. $\min\{-106/11, -30/11\} = -106/11, J_1 = \{4\}$, $q = 4$.
12. $B_1 = \{4, 5, 1\}$, $B_2 = \{7\}$, $N = \{2, 3, 6\}$, $k = 4$.

$$13.\ \text{Permuted } L = \begin{bmatrix} -1/2 & -2/7 & 1 \\ -1/4 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}, U = \begin{bmatrix} -8 & -4 & 5 \\ 0 & 7 & 5/4 \\ 0 & 0 & 41/7 \end{bmatrix}.$$

14. $\bar{c}_{B_2} = [-30/11]^T + (-(-106/11)/(41/22))[14/11]^T = [158/41]$
15. $k > m$, $\alpha > 0$.

Iteration 6:

4. $-B_2\bar{c}_{B_2} = -[5, 4, -3]^T[158/41] = [-790/41, -632/41, 474/41]^T$,
$w = [474/41, -1027/82, -4898/287]$,
$\Delta x_B = [-729/277, -2133/1681, -2905/997, 158/41]^T \neq 0$.
5. $J = \{7\} \neq \emptyset$.
6. $\alpha = \min\{335/10627\}$, $J_\alpha = \{7\}$.
7. $\bar{x}_B = [743/1731, 2007/964, 7204/1705, 233/1918]^T - (335/10627)$
$[-729/277, -2133/1681, -2905/997, 158/41]^T = [12244/23905, 87/41, 177/41, 0]^T$.
8. $J_\alpha \in B_2, B_1 = \{4, 5, 1\}; B2 = \emptyset$, $N = \{2, 3, 6, 7\}, k = 3 = m$.
16. $c_{B_1} = [0, -4, -6]^T$, $w = [0, -4/7, -37/41]^T$, $\bar{y} = [-37/41, -34/41, -27/41]^T$.

$$\bar{c}_N = \begin{bmatrix} 5 \\ -3 \\ 9 \\ -2 \end{bmatrix} - \begin{bmatrix} -2 & 5 & -3 & 5 \\ -1 & 6 & -5 & 4 \\ 3 & -2 & 1 & -3 \end{bmatrix}^T \begin{bmatrix} -37/41 \\ -34/41 \\ -27/41 \end{bmatrix} = \begin{bmatrix} 178/41 \\ 212/41 \\ 115/41 \\ 158/41 \end{bmatrix} \geq 0.$$

17. $T = \emptyset$. Terminated with primal and dual optimal solutions
$x^* = [177/41, 0, 0, 12244/23905, 87/41, 0, 0]^T$,
$y^* = [-37/41, -34/41, -27/41]^T$, $z* = [0, 178/41, 212/41, 0, 0, 115/41, 158/41]^T$.
Optimal value: $f^* = -1410/41$.
Phase-II ended in two iterations: one full iteration and one simple iteration each.

## 10. Final Remarks

First of all, implemented using the LU factorization, Algorithm 9.1 is suitable for sparse computing, the same as the simplex algorithm. Secondly, it seems to be stable, compared with the latter, since its pivot is selectable to some extent, while the pivot can be anywhere close to zero for the latter.

As for computational complexity, a simple iteration of Algorithm 9.1 is quite cheap, although it solves four triangular systems in a full iteration, as the simplex method. As a result, its computational effort in a single iteration is reduced on average.

To gain an insight into the behavior of the new face method, let us take a close look at its search direction $-\Delta x_B$.

Refer to subprogram (2.2) and its canonical form (4.7). Introduce notation

$$Z = \begin{bmatrix} -\bar{B}_2 \\ I_{k-m} \end{bmatrix} \tag{10.1}$$

It is evident that columns of $Z$ comprise a basis of the null of $B$. Denote the null by

$$V = \{p(v) \in R^k \mid p(v) = Zv, \quad v \in R^{k-m}\}.$$

We determine the "best" search direction for the following line search formula:

$$\hat{x}_B = \bar{x}_B - \alpha p(v), \qquad p(v) \in V.$$

The objective decrement, corresponding to $p(v) \in V$, can be obtained by combining (4.6) and (10.1), that is,

$$\begin{aligned} \delta(v) &= c_B^T \bar{x}_B - c_B^T \hat{x}_B = \alpha c_B^T p(v) \\ &= \alpha(\bar{c}_B^T + \bar{y}^T B)p(v) \\ &= \alpha \bar{c}_B^T p(v) \\ &= \alpha \begin{bmatrix} 0 \\ \bar{c}_{B_2} \end{bmatrix}^T \begin{bmatrix} -\bar{B}_2 \\ I_{k-m} \end{bmatrix} v \\ &= \alpha \bar{c}_{B_2}^T v \\ &= \alpha \|\bar{c}_{B_2}\|_2 \|v\|_2 \cos < \bar{c}_{B_2}, v > . \end{aligned}$$

In order to maximize $\delta(v)$ with $\alpha$ fixed and $v$ bounded, it is reasonable to find a such $v$ that solves the following program:

$$\begin{aligned} \max \quad & \delta(v) = \alpha \|\bar{c}_{B_2}\|_2 \|v\|_2 \cos < \bar{c}_{B_2}, v > \\ \text{s.t.} \quad & \|v\|_2 = \|\bar{c}_{B_2}\|_2, \ v \in R^{k-m}. \end{aligned}$$

It is clear that $v^* = \bar{c}_{B_2}$ is its optimal solution. Therefore, the associated vector, $-p(v^*) = -Z\bar{c}_{B_2}$, is a desirable choice for being the search direction. In fact, $p(v^*)$ is just $\Delta x_B$ defined by (4.10). As a result, the corresponding optimal value is $\delta(v^*) = \alpha \|\bar{c}_{B_2}\|_2^2$.

It is then seen that the amount, revealed by Theorem 5.3, is noting but the maximal objective decrement, obtained using $-\Delta x_B$ as the search direction, among all vectors in

the null of $B$. Therefore, one might as well think of $-\Delta x_B$ as a kind of *steepest downhill*, making Algorithm 9.1 very fast, in principle. Our experience with a number of small instances is consistent with this, although limited to length, this article only lists two.

At this stage, we cannot talk much about actual performance, but merely share some feelings. It seems to be a good sign that leaving indices in all the instances never re-enter the face thereafter, which would foretell the merit of the algorithm, though entering indices may re-leave. In fact, every instance was solved at the end of a single contraction process. An interesting question is whether this is the case in general. We leave computational experiments for the future.

## References


[1] E.M.L. Beale, Cycling in the dual simplex algorithm, *Naval Research Logistics Quarterly*, **2** (1955), 269-275.

[2] G.B. Dantzig, *Linear Programming and Extensions*, Princeton University Press, Princeton, New Jersey, 1963.

[3] G.H. Golub, Numerical methods for solving linear least squares problems,*Numer.Math.*, **7** (1965), 206-216.

[4] P.-Q. Pan, A -deficiency-allowing variation of the simplex method, *Computers and Mathematics with Applications*, **36** (1998) No. 3, 33-53.

[5] P.-Q. Pan, A projective simplex method for linear programming, *Linear Algebra and Its Applications*, **292** (1999), 99-125.

[6] P.-Q. Pan, A projective simplex algorithm using LU factorization, *Computers and Mathematics with Applications*, **39** (2000), 187-208.

[7] P.-Q. Pan, A primal deficient-Basis algorithm for linear programming, *Applied Mathematics and Computation*, **198** (2008b), 898-912.

[8] P.-Q. Pan, An affine-scaling pivot algorithm for linear programming, *Optimization*, **62** (2013), 431-445.

[9] P.-Q. Pan, *Linear Programming Computation*, Springer, Berlin Heidelberg, Germany, 2014.

[10] P.-Q. Pan, On "On an efficient implementation of the face algorithm for linear programming ", *Operations Research Transactions*, **19** (2015) No.3, 78-84.

[11] P.-Q. Pan, Revised face algorithm for linear programming, preprint, 2018.

[12] Y.-y Shi, L.-H. Zhang and W.-X. Zhu, A review of Linear Programming Computation by Ping-Qi Pan, *European Journal of Operational Research*, **267** (2018) No. 3, 1182-1183.

[13] L.-H. Zhang, W.-H. Yang and L.-Z. Liao, On an efficient implementation of the face algorithm for linear programming, *Journal of Computational Mathematics*, **31** ( 2013) No.4, 335-354.


School of Mathematics, Southeast University, Nanjing, 210096, China