A NEW FACE ALGORITHM USING LU FACTORIZATION FOR LINEAR PROGRAMMING

PING-QI PAN

SOUTHEAST UNIVERSITY

Abstract. The unique feature of the face algorithm [16] is that it moves from face to face, rather than from vertex to vertex as the simplex algorithm. It uses the orthogonal projection of the negative objective gradient on the related null space as its search direction. Nevertheless, the algorithm is based on QR factorization, which would not be very amenable for large sparse problems. In this paper, we present a face algorithm using LU factorization instead. It not only retains the main advantages of the original face algorithm, but also adds some nice features.

1. Introduction

We are concerned with the following linear programming (LP) problem:

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

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

The philosophy of the simplex algorithm [5] for solving LP problems is to move from vertex to vertex, until reaching an optimal vertex, unless detecting lower unboundedness.

As early as emergence of LP, degeneracy was found to be a problem for the simplex algorithm. A few instances were constructed that cannot be solved because of cycling.

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

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

What occurs is that the solution process falls into some degenerate vertex, and cannot get out of it forever [1, 7]. Although this rarely happens in practice, the process often pauses at a degenerate vertex for too long a time before exiting. In the academic community, there has been a common belief that degeneracy seriously degrades the performance of the simplex algorithm in solving large sparse LP problems.

In the past, great efforts have been made to get rid of degeneracy, in theory as well as in practice. As a recipe, a small perturbation term is added to the right-hand side to change column selection [3]. Some finite rules were also designed, such as those proposed in [2, 8]. Although there is no risk of cycling in theory, these rules cannot compete with Dantzig's original rule in practice.

Another recipe, so-called "deficient basis", came out twenty years ago [9, 14]. It is favorable that the related systems solved in each iteration are smaller, and the associated solution has fewer zero basic components, in general. Several variants were published with remarkable computational results [10]-[15]. It appears that the deficient-basis is particularly applicable to highly degenerate large-scale LP problems.

Nevertheless, the "fly in the ointment" is that columns of the deficient basis may increase in solution process. This leads us to believe that degeneracy is inextricable for the existing methodology that moves from one vertex to another.

The face algorithm, first proposed in [16], is a natural development in this regard. Its unique feature is that it no longer moves from vertex to vertex, but from face to face. Therefore, it brings hope of an escape from the degeneracy nightmare. And what's fascinating is that it provides an optimal face, rather than just a single optimal solution, opening up new possible applications (Section 25.2, [16]). Computational results are

preliminary but encouraging: the related dense code outperformed RSA, a simplex code, with time ratio of 4.91 on a set of 25 smallest standard Netlib problems.

Nevertheless, the face algorithm handles the involved normal system by QR factorization, which may not be very amenable for large sparse LP problems, even though its "steepest descent direction" is preferable. To make up for this weakness, Zhang, Yang and Liao [21] proposed an interesting variant of the algorithm, but it turned out to be unstable [17].

In this paper, we offer a new face algorithm based on LU factorization. It not only retains the main features of the original algorithm, but adds attractive features. In Section 2, we review the original face algorithm first. In Section 3, we derive a new search direction. Then, in Section 4, we discuss update of the feasible solution. Section 5 is devoted to pivoting operations and related face set updating. The theme of Section 6 is about optimality conditions. We formulate the new face algorithm in tableau form in Section 7, and in revised form in Section 8. Finally, we make remarks in Section 9.

In this paper, a_j denotes the j-indexed column of matrix A, and so on. For simplicity, the same notation for every matrix is also used for the index set of its columns. As usual, I_j denotes $j \times j$ unit matrix and e_j denotes unit vector with its j-th component 1.

2. Review of the original face algorithm

In this Section, we briefly review the original face algorithm and basic concepts, particularly focusing on the search direction used.

Partition A into (B, N), where $B \in \mathcal{R}^{m\times k}$, rank $B = m$, $N \in \mathcal{R}^{m\times(n-k)}$, $m \le k \le n$. Let $\bar{x}$ be a feasible solution to (1.1).

Using the preceding notation, we introduce the following concept.

Definition 2.1. *Associated with B, face is a nonempty feasible point (solution) set defined as*

$$P_B = \{(x_B, \bar{x}_N) \mid Bx_B + N\bar{x}_N = b,\ l_B \le x_B \le u_B;\ \bar{x}_j = l_j \text{ or } u_j,\ \forall j \in N\}. \tag{2.1}$$

B is termed *face matrix*, and N *nonface matrix*. Since $x_N = \bar{x}_N$ is fixed, N is also said *inactive*.

In particular, if $k = m$, P_B is 0-dimensional face, a vertex; if $k = n$, it is $(n - m)$-dimensional face, the feasible region itself. Thereafter, $k > m$ is assumed except indicated otherwise.

Any point (solution) in P_B is called *face point (solution)*. In the sequel, $\bar{x} \in P_B$ and its subvector $\bar{x}_B$ will not be distinguished.

Definition 2.2. *A face is level face if the objective value is constant over it. It is optimal face if the constant is equal to the optimal value.*

At the beginning of each iteration, we are faced with a face point $\bar{x}_B$ and a related subprogram:

$$\begin{aligned} \min \quad & c_B^T x_B + c_N^T \bar{x}_N \\ \text{s.t.} \quad & Bx_B + N\bar{x}_N = b, \quad l_B \le x_B \le u_B \end{aligned} \tag{2.2}$$

which minimizes the objective function over face P_B.

The orthogonal projection matrix onto the null space of B is known as

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

and hence the projection of the objective gradient c_B is

$$r = Pc_B = c_B - B^{\mathrm{T}}y, \qquad y = (BB^T)^{-1}Bc_B. \tag{2.3}$$

Vector $-r$ is our favorite search direction, since it is the "steepest downhill" in the null, with respect to the objective of (2.2).

In case when r vanishes, a level face is reached, and optimality is tested. If optimality condition is not fulfilled, related index sets are adjusted, and the next iteration is carried out (chapter 22, [16])

As for how to compute (y, r), defined by (2.3), the original algorithm [16] turns to the *normal system* below:

$$BB^{\mathrm{T}}y = Bc_B. \tag{2.4}$$

To solve, it uses Cholesky factorization of BB^T. The Cholesky factor is initially obtained from QR factorization of B^T, and updated iteration by iteration subsequently. A later variant [18] also handles the normal system, but it initiates and updates the inverse of BB^T instead.

On the other hand, y is just the unique solution to the least squares problem, stemming from the dual problem of (2.2), i.e.,

$$\min \|c_B - B^T y\|_2. \tag{2.5}$$

and r is the corresponding residual. This opens another door to (y, r).

One idea along this line, for instance, is to premultiply both c_B and B^T by an orthogonal matrix Q to convert the lease squares problem to

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

If Q is determined such that QB^T is upper triangular, it is then straightforward to compute (y, r).

As was mentioned, proposed was also another variant which can be implemented using LU factorization, but it failed because of instability. The present work is a new attempt to break the barriers.

3. Search direction

The quality of the search direction used is crucial to any LP solver. In this section, we derive a fresh new search direction.

The trick is to partition face matrix B further to

$$B = (B_1, B_2), \tag{3.1}$$

where $B_1 \in \mathcal{R}^{m\times m}$ with rank $(B_1) = m$ and $B_2 \in \mathcal{R}^{m\times(k-m)}$ $(m \le k \le n)$. B_1 and B_2 are called *basis (matrix)* and *active matrix*, respectively. Associated items, such as indices, variables etc, are said similarly.

Viewing the objective value f as a variable, we put subprogram (2.2) into the system of equations (leaving out bounds), expressed as the following tableau:

(3.2)

x_{B_1}	x_{B_2}	$\bar{x}_N$	f	RHS
B_1	B_2	N		b
$c_{B_1}^{\mathrm{T}}$	$c_{B_2}^{\mathrm{T}}$	c_N^T	-1	

where entries under $\bar{x}_N$ correspond to inactive (fixed) part.

Execute Gauss-Jordan elimination to convert the preceding tableau into the following form:

(3.3)

x_{B_1}	x_{B_2}	$\bar{x}_N$	f	RHS
I_m	$\bar{B}_2$	$\bar{N}$		$\bar{b}$
	$\bar{c}_{B_2}^T$	$\bar{c}_N^T$	-1	μ

which is termed *(canonical) face tableau*

Assuming $\bar{c}_{B_2} \neq 0$, we introduce the following simple result first.

Lemma 3.1. *For $k \times (k-m)$ matrix*

$$Z = \begin{pmatrix} -\bar{B}_2 \\ \\ I_{k-m} \end{pmatrix}, \tag{3.4}$$

it holds that

$$(I_m, \bar{B}_2)Z = 0, \qquad \text{and} \qquad rank(Z) = k - m. \tag{3.5}$$

Proof. The validity can be easily verified. □

Z is known as a *null space matrix* of $(I_m, \bar{B}_2)$, which is the coefficient matrix of the equality constraints of face tableau (3.3). Thus, the null space of $(I_m, \bar{B}_2)$ comprises of Zu, $\forall\, u \in R^{k-m}$.

In particular, consider the following vector from the null space:

$$\Delta x_B = Z\bar{c}_{B_2} = \begin{pmatrix} -\bar{B}_2\bar{c}_{B_2} \\ \\ \bar{c}_{B_2} \end{pmatrix}. \tag{3.6}$$

It is noted that $\Delta x_B \neq 0$ if and only if $k > m$ and $\bar{c}_{B_2} \neq 0$.

Theorem 3.1. *If Δx_B defined by (3.6) is nonzero, it holds that*

$$(I_m, \bar{B}_2)\Delta x_B = 0, \tag{3.7}$$

$$(0, \bar{c}_{B_2}^T)\Delta x_B > 0. \tag{3.8}$$

Proof. Validity of (3.7) is easily verified by (3.5) and (3.6).

In addition, it holds by (3.6) that

$$(0, \bar{c}_{B_2}^T)\Delta x_B = \bar{c}_{B_2}^T \bar{c}_{B_2} > 0,$$

which ends the proof. □

The preceding implies that nonzero $-\Delta x_B$ is downhill in the null space of $(I_m, \bar{B}_2)$, and hence is eligible to be taken as search direction. It is more than that. In the next Section, we will show that it is actually the best, in certain sense.

4. Update of the face solution

Let $\bar{x}_B$ be current face solution and $-\Delta x_B$ nonzero search direction available.

To update $\bar{x}_B$, we use the normal line search scheme, i.e.,

$$\hat{x}_B = \bar{x}_B - \alpha \Delta x_B, \tag{4.1}$$

where α is a step-size.

To be feasible, new solution $\hat{x}_B$ must satisfy

$$l_B \leq \hat{x}_B \leq u_B.$$

This restriction leads to the largest possible step-size determined, i.e,

$$\alpha = \min \alpha_j, \tag{4.2}$$

where

$$\alpha_j = \begin{cases} (\bar{x}_j - u_j)/\Delta x_j & \text{if } \Delta x_j < 0 \\ (\bar{x}_j - l_j)/\Delta x_j & \text{if } \Delta x_j > 0 \end{cases} \qquad j \in B \tag{4.3}$$

It is noted that α is nonnegative. In case of $\alpha = 0$, the so-called "new" solution determined by (4.1) is actually the same as the old. Such a solution $\bar{x}$ is said *degenerate*.

Theorem 4.1. *Assume that Δx_B defined by (3.6) is nonzero. If $\bar{x}_B$ is a face solution, so is $\hat{x}_B$ given by (4.1) along with (4.2) and (4.3) .*

Proof As a face solution, $\bar{x}_B$ satisfies the first m equations of (3.3), i.e.,

$$(I_m, \bar{B}_2)\bar{x}_B + N\bar{x}_N = b,$$

from which, (3.7) and (4.1), it follows that

$$(I_m, \bar{B}_2)\hat{x}_B + N\bar{x}_N = (I_m, \bar{B}_2)\bar{x}_B - \alpha(I_m, \bar{B}_2)\Delta x_B + N\bar{x}_N = (I_m, \bar{B}_2)\bar{x}_B + N\bar{x}_N = b$$

On the other hand, for α determined by (4.2) with (4.3), it can be verified that

$$l_B \le \bar{x}_B - \alpha\Delta x_B \le u_B$$

which along with (4.1) leads to $l_B \le \hat{x}_B \le u_B$. Therefore, $\hat{x}_B$ is a face solution. □

Now, let us explore the effect of the search direction further. Note that set $\{-Zu \mid u \in R^{k-m}\}$ includes all possible search direction candidates.

For any given $u \in R^{k-m}$, the associated update of $\bar{x}_B$ is

$$\hat{x}_B(u) = \bar{x}_B - \alpha Zu,$$

where α is step-size. Premultiplying the preceding by $(0, \bar{c}_{B_2}^T)$ and moving terms, we obtain the corresponding objective decrement, i.e,

$$\gamma(u) = (0, \bar{c}_{B_2}^T)\bar{x}_B - (0, \bar{c}_{B_2}^T)\hat{x}_B(u) = \alpha(0, \bar{c}_{B_2}^T)Zu, \tag{4.4}$$

which is not only depends on the direction of u but the norm of it. It is therefore reasonable to restrict u by

$$\|u\|_2 = \|\bar{c}_{B_2}\|_2. \tag{4.5}$$

Using (4.4) and (4.5) , we then turn to the following program:

$$\begin{array}{ll} \max & \gamma(u) = \alpha(0, \bar{c}_{B_2}^T)Zu \\ \text{s.t.} & \|u\|_2 = \|\bar{c}_{B_2}\|_2, \qquad u \in R^{k-m}. \end{array} \tag{4.6}$$

Theorem 4.2. *Program (4.6) has unique optimal solution $u^* = \bar{c}_{B_2}$, corresponding to maximum $\gamma(u^*) = \alpha\|\bar{c}_{B_2}\|_2^2$.*

Proof. From (3.4), (4.4) and the constraint of (4.6), it follows that

$$\begin{aligned} \gamma(u) &= \alpha(0, \bar{c}_{B_2}^T) \begin{pmatrix} -\bar{B}_2 \\ I_{k-m} \end{pmatrix} u \\ &= \alpha\bar{c}_{B_2}^T u \\ &= \alpha\|\bar{c}_{B_2}\|_2\|u\|_2 \cos < \bar{c}_{B_2}, u > \\ &= \alpha\|\bar{c}_{B_2}\|_2^2 \cos < \bar{c}_{B_2}, u >, \end{aligned} \tag{4.7}$$

which implies that problem (4.6) has the unique optimal solution $u^* = \bar{c}_{B_2}$, corresponding to maximum $\gamma(u^*) = \alpha\|\bar{c}_{B_2}\|_2^2$. □

Theorem 4.2 ensures that $\gamma(u^*)$, being proportional to the square of $\|\bar{c}_{B_2}\|_2$, is the biggest possible objective decrement that can be achieved. Therefore, search direction $-\Delta x_B = -Z\bar{c}_{B_2}$ appears to be the "best" choice.

Unfortunately, the objective decrement also depends on step-size α. It even vanishes if so is α. It seems to be reasonable to take variables change into account.

To this end, consider the maximum possible range of variation that can occur in $\bar{x}_j \in B_2$, i.e.,

$$w_j = \begin{cases} u_j - \bar{x}_j & \text{if} \quad \bar{c}_j < 0 \\ 0 & \text{if} \quad \bar{c}_j = 0 \\ \bar{x}_j - l_j & \text{if} \quad \bar{c}_j > 0 \end{cases} \qquad j \in B_2 \tag{4.8}$$

Note that $w_j \geq 0,\ \forall j \in B_2$. In case of $\bar{c}_j = 0$, w_j could be set freely, as any change in $\bar{x}_j$ doesn't matter with the objective value; it is set so any way that $\bar{x}_j$ stays the same.

Practically, it might be somehow attractive to use (3.6) with $u^* = \bar{c}_{B_2}$ replaced by $u = W\bar{c}_{B_2}$, i.e.,

$$\Delta x_B = Z(W\bar{c}_{B_2}) = \begin{pmatrix} -\bar{B}_2 W\bar{c}_{B_2} \\ \mathrm{W}\bar{c}_{B_2} \end{pmatrix}, \tag{4.9}$$

where W is diagonal matrix with $w_j \in B_2$ as diagonal entries.

Further, considering that w_j could be too large, we may use the following normalization instead:

$$w = w/\|w\|_\infty, \tag{4.10}$$

where w denotes the vector consisting of w_j, $j \in B_2$.

5. Pivoting operation

In this section, we describe pivoting operation and related index set updating.

Assume that a new face solution $\hat{x}_B$ has been computed. Then, the following index set is well-defined:

$$J' = \arg\min_{j\in B} \alpha_j, \tag{5.1}$$

where α_j is determined by (4.3). It is clear that for each $j \in J'$, $\hat{x}_j$ reaches one of its bounds. Such indices and related variables are known as *blocking* ones.

There are the following two types of iterations.

(A) Simple iteration: $\tilde{J} \stackrel{\triangle}{=} B_2 \cap J' \neq \emptyset$ (or $J' \not\subset B_1$).

In this case, there is no need for basis change, and thus the involved computational cost is quite cheap. Before carrying out the next iteration, what to do is only to perform the following tactic, contracting the face:

Tactic 5.1. *update* (B_2, N, k) *by*

$$B_2 = B_2 \backslash \tilde{J}, \qquad N = N \cup \tilde{J}, \qquad k = k - |\tilde{J}|. \tag{5.2}$$

(B) Full iteration: $\tilde{J} \stackrel{\triangle}{=} B_2 \cap J' = \emptyset$ (or $J' \subset B_1$).

This time, there is a need for pivot selection and basis change. Note that the i-th basic variable x_{j_i} corresponds to the i-th row of tableau (3.3).

Determine pivot row by the following rule first.

Rule 5.1. *[Row Rule] Assume that Δx_B and J' are defined by (3.6) and (5.1), respectively. Select row index p such that*

$$j_p \in \arg\max_{j_i \in J'} |\Delta x_{j_i}|. \tag{5.3}$$

Once blocking index $j_p \in B_1$ is determined to leave B_1, what to do next is to select an entering index from B_2. Among multiple possible choices, we bring up the following plausible one.

Rule 5.2. *[Column Rule] Assume that Δx_B and j_p are defined by (3.6) and (5.3), respectively. Select entering index q such that*

$$q \in \begin{cases} \arg\max\{v_j \bar{c}_j \mid j \in B_2\}, & \text{if} \quad \Delta x_{j_p} < 0, \\ \arg\min\{v_j \bar{c}_j \mid j \in B_2\}, & \text{if} \quad \Delta x_{j_p} > 0, \end{cases} \tag{5.4}$$

where $v_{B_2}^T$ denotes the p-th row of $\bar{B}_2$.

Consequently, $-v_q \bar{c}_q$ coincides in sign with Δx_{j_p}. It is obvious that a nonzero pivot v_q is then ensured. But it is more than that, since the rule conforms to the heuristic characteristic of the optimal solution (section 2.5, [16]).

Then, we update the canonical face tableau (3.3), just as in the simplex context, and update sets by the following tactic.

Tactic 5.2. *For full iteration, update (B_1, B_2, N, k) by*

$$B_1 = (B_1 \backslash \{j_p\}) \cup \{q\},\ B_2 = B_2 \backslash \{q\},\ N = N \cup \{j_p\},\ k = k - 1. \tag{5.5}$$

It is noted that the blocking index j_p was then kicked into inactive set N.

6. Optimality condition

It is noted that neither (3.6) nor (4.9) is well-defined if $k = m$. Otherwise, the search direction vanishes if $k > m$ but $\bar{c}_{B_2} = 0$. In these cases, it is time to test for optimality.

Lemma 6.1. *Let (3.3) be a face tableau. If $k = m$, or $k > m$ but $\bar{c}_{B_2} = 0$, it corresponds to a level face.*

Proof. Note that the tableau corresponds to face P_B with objective function

$$f = -\mu + \bar{c}_{B_2}^T x_{B_2} + \bar{c}_N^T \bar{x}_N. \tag{6.1}$$

If $k = m$ and hence $B_2 = \emptyset$, it is clear that the associated feasible solution is a basic solution, and P_B is hence a special level face, vertex.

If $k > m$ and $\bar{c}_{B_2} = 0$, equation (6.1) becomes

$$f = -\mu + \bar{c}_N^T \bar{x}_N, \tag{6.2}$$

which means that the objective value is the same over $x \in P_B$. Therefore, P_B is a level face. □

Theorem 6.1. *Let (3.3) be a face tableau with $k = m$, or $k > m$ but $\bar{c}_{B_2} = 0$. If the following index set is empty:*

$$T = \{j \in N \mid \bar{x}_j = l_j,\ \bar{c}_j < 0, \} \cup \{j \in N \mid \bar{x}_j = u_j,\ \bar{c}_j > 0\}, \tag{6.3}$$

the tableau gives an optimal face together with optimal solution $\bar{x}$ to problem (1.1).

Proof. Since the current $\bar{x}$ is a feasible solution to (1.1), we only need to check coefficients of the objective row of the tableau. Note that $\bar{c}_{B_1}$ vanishes, and $\bar{x}_j$ is on one of its bounds for all $j \in N$.

Without loss of generality, assume that $k > m$. Under the assumption, $\bar{c}_{B_2} = 0$, and T is empty, which implies that

$$\begin{array}{lll} \bar{c}_j \geq 0 & \text{if} & \bar{x}_j = l_j, \\ \bar{c}_j \leq 0 & \text{if} & \bar{x}_j = u_j. \end{array} \qquad \forall\ j \in N \tag{6.4}$$

Therefore, there is no feasible solution other than $\bar{x}$, which corresponds to a lower objective value. Thus, it is shown that $\bar{x}$ is an optimal solution.

By Lemma 6.1, the tableau corresponds to a level face. Since it includes $\bar{x}$, the face is actually optimal. □

If $T \neq \emptyset$, optimality cannot be claimed. In this case, we execute the following tactic, and carry out the next iteration.

Tactic 6.1. *To expand the face, update* (B_2, N, k) *by*

$$B_2 = B_2 \cup T, \qquad N = N \backslash T, \qquad k = k + |T|. \tag{6.5}$$

7. Face algorithm: tableau form

Based on discussions made in previous sections, we formulate a new face algorithm in tableau form in this section.

Assume that $\bar{x}$ is an initial feasible solution. Let B_1 be an $m \times m$ nonsingular submatrix of A. Set

$$N = A \backslash B_1, \qquad B_2 = \emptyset.$$

and generate an initial face tableau in form (3.3).

In each iteration, $\bar{x}_B$ is updated with (B_1, B_2, N) adjusted accordingly, so that the cardinality $|B_2| = k - m$ decreases by one, at least. A series of iterations are performed

until $k = m$ or $k > m$ but $\bar{c}_{B_2} = 0$ (unless step-size α is too large, and lower unboundedness should be declared). Such a series is referred to as *face contraction*. At the end of it, optimality is tested. If set T is empty, optimality is achieved; otherwise, the next face contraction is carried out.

Overall steps are summarized into the following model.

Algorithm 1. [Face algorithm: tableau form] Initial : Partition B_1, $N = A\backslash B_1$, $B_2 = \emptyset$, $k = m$; face tableau in form (3.3); feasible solution $\bar{x}$; big number σ. This algorithm solves problem (1.1).

1. Stop if T defined by (6.3) is empty.
2. Adjust index sets by Tactic 6.1.
3. Compute Δx_B by (3.6).
4. Determine α by (4.2) with (4.3).
5. Stop if $\alpha > \sigma$.
6. If $\alpha > 0$, update $\bar{x}_B$ by (4.1).
7. If $J' \cap B_2 \neq \emptyset$, where J' is defined by (5.1), then:
   (1) update index sets by Tactic 5.1;
   (2) if $k = m$ or $\bar{c}_{B_2} = 0$, go to step 1; else to step 3.
8. Determine row index p and leaving index j_p by (5.3).
9. Select entering index q by (5.4).
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 $i = 1, \ldots, m+1;\ i \neq p$.
11. Update index sets by Tactic 5.2.
12. If $k = m$ or $\bar{c}_{B_2} = 0$, go to step 1; else to step 3.

Theorem 7.1. *If each face contraction involves a non-degenerate feasible solution, Algorithm 1 terminates either at*

(i) step 1, giving an optimal face together with an optimal solution; or at

(2) step 5, declaring lower unboundedness.

Proof. It is evident that Algorithm 1 reaches a level face at the end of each face contraction within finitely many iterations, unless it exits from step 5, declaring lower unboundedness.

If the algorithm does not terminate, then it performs infinitely many face contractions. Under the non-degeneracy assumption, the sequence of level faces reached by these contractions correspond to strictly decreasing objective values (Theorem 4.2), which implies that there are infinitely many level faces, a contradiction. Therefore, the algorithm terminates.

By Theorem 6.1, exiting from step 1 gives an optimal face together with an optimal solution. □

8. Face algorithm: revised form

Assume that $\bar{x}$ and $\bar{c}_{B_2}$ are current feasible solution and active reduced cost, respectively. Let the following LU factorization be available:

$$B_1 = LU.$$

The following key quantities can be derived with equality $\bar{B}_2 = B^{-1}B_2$:

(a) Search direction $\Delta x_B = (\Delta x_{B_1}, \Delta x_{B_2})$ defined by (3.6).

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

$$Lu = -B_2 \bar{c}_{B_2}, \qquad U \Delta x_{B_1} = u. \tag{8.1}$$

(b) Update of active reduced cost $\bar{c}_{B_2}$:

$$\hat{c}_{B_2} = \bar{c}_{B_2} + \beta v_{B_2}, \qquad \beta = -\bar{c}_q / v_q, \tag{8.2}$$

where q is the entering index, and $v_{B_2}^T$ is the p-th row of $\bar{B}_2$.

To obtain $v_{B_2} = B_2^T B_1^{-T} e_p$, solve the following $m \times m$ triangular systems for $t = B_1^{-T} e_p$ first:

$$U^T s = e_p, \qquad L^T t = s, \tag{8.3}$$

and then compute

$$v_{B_2} = B_2^T t. \tag{8.4}$$

Now we are able to transform Algorithm 1 into the following model.

Algorithm 2. [Face algorithm: revised form] Initial : Given initial partition $B_1, N = A \backslash B_1, B_2 = \emptyset$, $k = m$; factorization $B_1 = LU$; feasible solution $\bar{x}$ and active reduced cost $\bar{c}_{B_2}$; big number σ. This algorithm solves problem (1.1).

1. Stop if T defined by (6.3) is empty (optimality achieved).
2. Adjust index sets by Tactic 6.1.
3. Set $\Delta x_{B_2} = \bar{c}_{B_2}$, and solve triangular systems (8.1) for Δx_{B_1}.
4. Determine α by (4.2) with (4.3).
5. Stop if $\alpha > \sigma$ (lower unboundedness declared).
6. If $\alpha > 0$, update $\bar{x}_B$ by (4.1).

7. If $J' \cap B_2 \neq \emptyset$, where J' is defined by (5.1), then:

   (1) update index sets by Tactic 5.1;

   (2) if $k = m$ or $\bar{c}_{B_2} = 0$, go to step 1; else to step 3.

8. Solve triangular systems (8.3), and compute v_{B_2} by (8.4).

9. Determine row index p and leaving index j_p by (5.3).

10. Select entering index q by (5.4).

11. Update L and U.

12. Update index sets by Tactic 5.2.

13. If $k > m$, update $\bar{c}_{B_2}$ by (8.2).

14. If $k = m$ or $\bar{c}_{B_2} = 0$, go to step 1; else to step 3.

9. Final remarks

The new face algorithm has the following features:

At first, unlike the simplex algorithm, which uses the search direction along some descent edge of the underlying polyhedron, the new algorithm uses a combination of all edges corresponding to nonzero active reduced costs.

Secondly, as a result, the new algorithm no longer moves from vertex to vertex, but from face to face, with a sequence of corresponding face points, until reaching an optimal face together with an optimal point. Therefore, it would have more chance to bypass degenerate vertices.

Thirdly, the new algorithm is stable, compared with simplex algorithms, as its pivot is selectable, to some extent, while the latter's can be arbitrarily close to zero. As we all know, the latter sometimes fails to maintain a usable basis, and has to restart from scratch.

As for computational cost in a single iteration, the new algorithm is about the same as simplex algorithms. It solves four triangular systems, just as the latters. Using LU factorization, the former can be implemented, as the same as in the conventional simplex context.

Finally, the unique and promising feature of the new algorithm is the maximum objective decrement $\alpha\|\bar{c}_{B_2}\|_2^2$ achieved in a single iteration. We expect that the number of iterations required could be reduced significantly. At this stage, however, we cannot talk much about its actual performance. Concerning search directions (3.6) and (4.9), e.g., we don't know which one is better although we used the former. Alternatively, instead of the whole $\bar{c}_{B_2}$, it is possible to use a part of it to form search direction, e.g., such that

$$|\bar{c}_j| \geq \gamma, \qquad j \in B_2, \tag{9.1}$$

where γ is a threshold decreasing in solution process. All of these need to be considered. The implementation of the new algorithm and computational experiments are expected.

Author: Ping-Qi Pan, Professor of School of Mathematics, Southeast University, Nanjing 210096, China. Email: panpq@seu.edu.cn

Funding: This study was funded by National Natural Science Foundation of China (grant number 10871043)

Conflict of Interest: The authors declare that they have no conflict of interest

References

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

[2] R.G. Bland, New finite pivoting rules for the simplex method,*Mathematics of Operations Research* , **2** (1977), 103-107.

[3] A. Charnes, Optimality and degeneracy in linear programming,*Econometrica* , **20** (1952), 160-170.

[4] G.B. Dantzig, Programming in a linear structure, Comptroller, USAF, Washington, D.C.(February 1948).

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

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

[7] A.J. Hoffman, Cycling in the simplex algorithm, Technical Report 2974,National Bureau of Standards, 1953.

[8] P.-Q. Pan, Practical finite pivoting rules for the simplex method, *OR Spektrum* , **12** (1990), 219-225.

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

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

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

[12] P.-Q. Pan, A dual projective pivot algorithm for linear programming, *Computational Optimization and Applications*, **29** (2004), 333-344.

[13] P.-Q. Pan, A revised dual projective pivot algorithm for linear programming, *SIAM Journal on Optimization*, **16** (2005), 49-68.

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

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

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

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

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

[19] M. A. Saunders, *Large Scale Linear Programming Using the Cholesky Factorization*, Stanford University Report STAN-CS-72-152, 1972.

[20] 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.

[21] 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.