Journal of Computational Mathematics, Vol.11, No.4, 1993, 350–364.

FURTHER DEVELOPMENTS IN AN ITERATIVE PROJECTION AND CONTRACTION METHOD FOR LINEAR PROGRAMMING*1)

He Bing-sheng
(*Nanjing University, Nanjing, China*)

Abstract

A linear programming problem can be translated into an equivalent general linear complementarity problem, which can be solved by an iterative projection and contraction (PC) method [6]. The PC method requires only two matrix-vector multiplications at each iteration and the efficiency in practice usually depends on the sparsity of the constraint-matrix. The prime PC algorithm in [6] is globally convergent; however, no statement can be made about the rate of convergence. Although a variant of the PC algorithm with constant step-size for linear programming [7] has a linear speed of convergence, it converges much slower in practice than the prime method [6]. In this paper, we develop a new step-size rule for the PC algorithm for linear programming such that the resulting algorithm is globally linearly convergent. We present some numerical experiments to indicate that it also works better in practice than the prime algorithm.

1. Introduction

This paper presents an algorithm for linear programming problems based on an iterative projection and contraction method for linear complementarity problems [6]. The algorithm makes a trivial projection onto a general orthant at each iteration and the generated sequence contracts Féjer-monotonically to the solution set, i.e., the Euclidean distance of the iterates to the solution set decreases at each iteration. Usually the matrices describing the constraints for large problems will be sparse, but often no special structure pattern is detectable in it. The projection and contraction method for linear complementarity problems is an iterative procedure which requires in each step only two matrix-vector multiplications, and performs no transformation on the matrix elements. The method therefore allows the optimal exploitation of the sparsity of the constraint matrices and may thus be an efficient method for large sparse problems ([6] and [7]).

* Received May 12, 1992.
1) The Project Supported by National Natural Science Foundation of China.

In this paper, we work on the ideas in [6] and [7], and obtain a faster algorithm. The directions generated by this algorithm are the same as generated by the algorithms presented in [6] and [7]. However, we get a new simple step-size rule and are able to obtain global linear convergence without estimation of the norm of the matrix describing the constraints. Moreover, the new algorithm also works better in practice than the prime algorithm.

Our paper is organized as follows. In Section 2, we quote some theoretical background from [6]. Section 3 describes the new algorithm and its relation to other PC algorithms. Section 4 proves the convergence properties of our new algorithm. Section 5 gives an extension–the scaled algorithm. In Section 6, we present some numerical results. Finally, in Section 7, we conclude the paper with some remarks.

We use the same notations as in [6]. The i-th component of a vector x in the real n-dimensional Euclidean space R^n is denoted by x_i. A superscript such as in u^k refers to specific vectors and k usually denotes the iteration index. $P_\Omega(\cdot)$ denotes the orthogonal projection on the convex closed set Ω. Specifically, x_+ denotes the projection of x on nonnegative orthant R^n_+, i.e.,

$$(x_+)_i := \max\{0, x_i\}, \qquad i = 1, \dots, n.$$

$\|\cdot\|$ and $\|\cdot\|_\infty$ denote the Euclidean and the max-norm, respectively. For a positive definite matrix G, the norm $\|u\|_G$ is given by $(u^TGu)^{\frac{1}{2}}$.

2. Theoretical Background

We consider the pair of the standard form linear program and its dual

$$\text{(P)} \qquad \begin{aligned} &\min \quad c^Tx, \\ &\text{s.t} \quad Ax = b, \quad x \geq 0, \end{aligned} \tag{1}$$

$$\text{(D)} \qquad \begin{aligned} &\max \quad b^Ty, \\ &\text{s.t} \quad A^Ty \leq c \end{aligned} \tag{2}$$

where A is an $m \times n$-matrix and b, c are vectors of length m and n, respectively. Let

$$\Omega^* := \{u = (x, y) | x \text{ is a solution of (P)}, \ y \text{ is a solution of (D)}\}. \tag{3}$$

Throughout the paper we assume that $\Omega^* \neq \emptyset$. It is well known [2] that $u = (x, y) \in \Omega^*$ if and only if it solves the following general linear complementarity problem:

$$\text{(LCP)} \qquad \begin{cases} x \geq 0, \quad -A^Ty + c \geq 0, \quad x^T(-A^Ty + c) = 0, \\ Ax - b = 0. \end{cases} \tag{4}$$

Let

$$M := \begin{pmatrix} & -A^T \\ A & \end{pmatrix}, \qquad q := \begin{pmatrix} c \\ -b \end{pmatrix}, \qquad \Omega := \{u = (x, y) | x \geq 0\}. \tag{5}$$

Then (4) is also characterized by

$$\text{(LCP)} \qquad \begin{cases} u_I \geq 0, \\ (Mu+q)_I \geq 0, \\ u_I^T(Mu+q)_I = 0, \\ (Mu+q)_{L\setminus I} = 0, \end{cases}$$

where $I = \{1, \ldots, n\}$ and $L = \{1, \ldots, n+m\}$. The projection $\tilde{u} = P_\Omega[u]$ of u onto Ω is simply given by

$$\tilde{x} = x_+, \qquad \tilde{y} = y.$$

It is easy to see (see e.g. Mangasarian [9]) that u^* is a solution of the LCP if and only if it is a zero of the function

$$e(u) := u - P_\Omega[u - (Mu+q)]. \tag{6}$$

We will measure the "distance" to such a zero by the function

$$\psi(u) := \|e(u)\|^2. \tag{7}$$

The relationship of a violation of the optimality conditions above and the function ψ can be illustrated by the following table in which we assume that only one of four conditions of the LCP is violated in each case:

$$\text{If} \quad \left\{ \begin{array}{l} u_I \leq 0 \\ (Mu+q)_I \leq 0 \\ u_I^T(Mu+q)_I > 0 \\ (Mu+q)_{L\setminus I} \neq 0 \end{array} \right\} \quad \text{then} \quad \psi(u) = \begin{cases} \|u_I\|^2, \\ \|(Mu+q)_I\|^2, \\ \|\min\{u_I, (Mu+q)_I\}\|^2, \\ \|(Mu+q)_{L\setminus I}\|^2. \end{cases}$$

Let

$$\varphi(u) := e(u)^T(Mu+q). \tag{8}$$

We have the following basic lemma.

Lemma 1. *Let $u \in \Omega$. Then*

$$\varphi(u) \geq \psi(u). \tag{9}$$

A simple proof of Lemma 1 can be found in [6]. From this result we obtain immediately the following

Theorem 1. *Let $\psi(u)$ and $\varphi(u)$ be defined as in* (7) *and* (8), *respectively. Then*

i) $\varphi(u) \geq \psi(u) \geq 0$ *for all* $u \in \Omega$,

ii) $u \in \Omega$ *and* $\varphi(u) = 0 \iff \psi(u) = 0 \iff u \in \Omega^*$.

For $u \in \Omega$, the functions $\psi(u)$ and $\varphi(u)$ are continuous and can be viewed as measures for the distance of u from the solution set Ω^*. Moreover, one can show that for LCP's with a unique solution u^* there exists a constant $\eta(M,q,r) > 0$ such that

$$\eta(M,q,r) \le \frac{\|e(u)\|}{\|u-u^*\|} \le \|M\| + 2$$

is true for all $u \in S := \{u \in \Omega \mid \|u - u^*\| \le r\}$. In our algorithm below we take the vector

$$g(u) := M^T e(u) + (Mu + q) \tag{10}$$

as the search direction. The following theorem plays an important role in the projection and contraction method.

Theorem 2. *Let $u \in \Omega$, $u^* \in \Omega^*$ and $g(u)$ be defined as above. Then we have*

$$(u-u^*)^T g(u) \ge \varphi(u). \tag{11}$$

The proof of Theorem 2 can also be found in [6]. For $u \in \Omega$, the direction $-g(u)$ is a profitable direction, i.e. a descent direction of $\|u-u^*\|^2$. As a consequence of Theorem 2, one can build a prime PC algorithm (see [1] and [6]), which is globally convergent whenever $\Omega^* \ne \emptyset$.

3. The New Algorithm and Its Relation to Other Algorithms

First, we state our new algorithm.

PC Algorithm 1.

Step 0. Assume Δ_1 and Δ_2 are fixed constants satisfying $0 < \Delta_1 \le \Delta_2 < 2$. Let $\varepsilon > 0$ and $u^0 \in \Omega$. Set $k := 0$.

Step 1. Compute $e(u^k)$. If $\dfrac{\|e(u^k)\|_\infty}{\|q\|_\infty} \le \varepsilon$, stop.

Step 2. Calculate $g(u^k)$. Set

$$\alpha_k = \frac{\|M^T e(u^k)\|^2}{\|e(u^k)\|^2}, \tag{12}$$

$$\rho_k = \frac{1}{1+\alpha_k} \tag{13}$$

and take

$$\gamma_k \in [\Delta_1, \Delta_2]. \tag{14}$$

Step 3. Set

$$\bar{u}^k = u^k - \gamma_k \rho_k g(u^k), \tag{15}$$

$$u^{k+1} = P_\Omega[\bar{u}^k]. \tag{16}$$

Set $k := k+1$ and go to Step 1.

The number γ_k is a factor for the step length, similar to ω of the SOR-method. For linear programming, the best choice of γ may be a number very close to 2. The main difference between our new algorithm and PC algorithms in [6] and [7] is that they take different step-size rules (how to choose ρ). For convenience, we fix $\gamma_k \equiv 1$. In the prime algorithm [6], for $u \in \Omega$, one denotes

$$N(x) := \{i \mid x_i = 0 \text{ and } (g_x)_i \geq 0\},$$

$$B(x) := \{1, \ldots, n\} \setminus N(x).$$

Correspondingly, denote

$$u = \begin{pmatrix} x_B \\ x_N \\ y \end{pmatrix}, \quad g(u) = \begin{pmatrix} g_{x_B} \\ g_{x_N} \\ g_y \end{pmatrix}, \quad g_B(u) = \begin{pmatrix} g_{x_B} \\ 0 \\ g_y \end{pmatrix}, \quad g_N(u) = \begin{pmatrix} 0 \\ g_{x_N} \\ 0 \end{pmatrix}.$$

It is easy to see that for $u \in \Omega$ and $u^* \in \Omega^*$,

$$(u - u^*)^T g_B(u) \geq (u - u^*)^T g(u) \geq \varphi(u).$$

The choice of the prime algorithm [6] is

$$\rho_{\text{prime}} = \frac{\varphi(u)}{\|g_B(u)\|^2}. \tag{17}$$

This choice guarantees that the generated sequence $\{u^k\}$ satisfies

$$\|u^{k+1} - u^*\|^2 \leq \|u^k - u^*\|^2 - \rho_{\text{prime}}\|e(u^k)\|^2$$

and converges globally to the solution set Ω^*. However, in the worst case, ρ_{prime} may be very small and the algorithm exhibits a slow convergence behaviour.

In the variant of the PC algorithm with constant step-size [7], one needs first an estimation of $\|A\|$, say $\beta \geq \|A\|$, and then takes

$$\rho_{\text{const}} = \frac{1}{1 + \beta^2} \tag{18}$$

as step-size in all iterations. The generated sequence $\{u^k\}$ satisfies

$$\|u^{k+1} - u^*\|^2 \leq \|u^k - u^*\|^2 - \rho_{\text{const}}\|e(u^k)\|^2$$

and the algorithm is globally linearly convergent since $\rho_k \equiv \rho_{\text{const}}$ is bounded below. Unfortuately, in practice this algorithm converges much slower than the prime one. In [7] we suggest choosing $\rho = \max\{\rho_{\text{prime}}, \rho_{\text{const}}\}$ so that ρ_{const} serves as a safeguard against a "too short" step length.

The step-size ρ_{new} in our new algorithm is chosen by

$$\rho_{\text{new}} = \frac{1}{1 + \alpha_k}$$

and α_k is calculated in every iteration by (12). In the next section we will prove that the sequence $\{u^k\}$ generated by this new algorithm satisfies

$$\|u^{k+1} - u^*\|^2 \le \|u^k - u^*\|^2 - \rho_{\text{new}}\|e(u^k)\|^2.$$

From (12) and (5)

$$\alpha_k \le \|M\|^2 = \|A\|^2 \le \beta^2.$$

It follows that

$$\rho_{\text{new}} \ge \rho_{\text{const}}. \tag{19}$$

Therefore the new choice of the step length may offer a faster algorithm with the same worst-case bound as the algorithm in [7]. We note that the projection on the general orthant Ω is trivial and in all PC algorithms the main work at each iteration is the computation of Mu and $M^Te(u)$. For linear programming, each iteration requires two matrix-vector multiplications of the form Ax and two multiplications of the form A^Ty.

4. Convergence Results

We begin this section by stating the main result.

Theorem 3. *The sequence $\{u^k\}$ generated by the PC Algorithm 1 for linear programs in standard form satisfies*

$$\|u^{k+1} - u^*\|^2 \le \|u^k - u^*\|^2 - \gamma_k(2-\gamma_k)\rho_k\|e(u^k)\|^2. \tag{20}$$

Proof. The proof of Theorem 3 is divided into two steps. First we let

$$\Theta_k = \|u^k - u^*\|^2 - \|u^{k+1} - u^*\|^2 - \gamma_k(2-\gamma_k)\rho_k\|e(u^k)\|^2 \tag{21}$$

and rewrite Θ_k in a convenient expression. Then we show the equivalent assertion $\Theta_k \ge 0$. In the sequel we shall write briefly $\rho,\ \gamma,\ u,\ \bar{u},\ \tilde{u}$ and Θ instead of $\rho_k,\ \gamma_k,\ u^k,\ \bar{u}^k,\ u^{k+1}$ and Θ_k. Since $\tilde{u} = P_\Omega(\bar{u})$ and $u^* \in \Omega$,

$$\|\bar{u} - u^*\|^2 \ge \|\tilde{u} - u^*\|^2 + \|\bar{u} - \tilde{u}\|^2, \tag{22}$$

and from (11) and (15) follows

$$\begin{aligned} \|u - u^*\|^2 - \|\tilde{u} - u^*\|^2 &\ge \|u - u^*\|^2 - \|\bar{u} - u^*\|^2 + \|\bar{u} - \tilde{u}\|^2 \\ &= 2\gamma\rho(u-u^*)^Tg(u) - \gamma^2\rho^2\|g(u)\|^2 + \|\bar{x} - \tilde{x}\|^2 \\ &\ge 2\gamma\rho\varphi(u) - \gamma^2\rho^2\|g(u)\|^2 + \|\bar{x} - \tilde{x}\|^2. \end{aligned} \tag{23}$$

It follows from (23) that

$$\Theta \ge 2\gamma\rho\varphi(u) - \gamma(2-\gamma)\rho\|e(u)\|^2 - \gamma^2\rho^2\|g(u)\|^2 + \|\bar{x} - \tilde{x}\|^2. \tag{24}$$

Denote

$$z := Ax - b, \qquad v := A^T y - c, \qquad w := -A^T z.$$

Then

$$Mu + q = \begin{pmatrix} -A^T y + c \\ Ax - b \end{pmatrix} = \begin{pmatrix} -v \\ z \end{pmatrix},$$

$$u - P_\Omega[u - (Mu+q)] = \begin{pmatrix} x - (x+v)_+ \\ z \end{pmatrix},$$

$$\varphi(u) = \{u - P_\Omega[u - (Mu+q)]\}^T (Mu+q) = [(x+v)_+ - x]^T v + z^T z, \tag{25}$$

$$\|e(u)\|^2 = \|u - P_\Omega[u - (Mu+q)]\|^2 = \|(x+v)_+ - x\|^2 + z^T z, \tag{26}$$

$$M^T\{u - P_\Omega[u - (Mu+q)]\} = \begin{pmatrix} -w \\ A[(x+v)_+ - x] \end{pmatrix},$$

$$g(u) = M^T\{u - P_\Omega[u - (Mu+q)]\} + (Mu+q) = \begin{pmatrix} -w - v \\ A[(x+v)_+ - x] + z \end{pmatrix}, \tag{27}$$

and

$$\|g(u)\|^2 = \|w+v\|^2 + \|A[(x+v)_+ - x]\|^2 + z^T z - 2[(x+v)_+ - x]^T w. \tag{28}$$

Note that

$$\alpha = \frac{\|M^T\{u - P_\Omega[u - (Mu+q)]\}\|^2}{\|u - P_\Omega[u - (Mu+q)]\|^2} = \frac{w^T w + \|A[(x+v)_+ - x]\|^2}{z^T z + \|(x+v)_+ - x\|^2}$$

and

$$\|A[(x+v)_+ - x]\|^2 = \alpha z^T z + \alpha\|(x+v)_+ - x\|^2 - w^T w. \tag{29}$$

It follows from (28) and (29) that

$$\|g(u)\|^2 = \|w+v\|^2 + (1+\alpha) z^T z + \alpha\|(x+v)_+ - x\|^2 - w^T w - 2[(x+v)_+ - x]^T w. \tag{30}$$

From (24), (25), (26) and (30) we obtain

$$\begin{aligned}\Theta &\ge 2\gamma\rho\{[(x+v)_+ - x]^T v + z^T z\} - (2\gamma - \gamma^2)\rho\{\|(x+v)_+ - x\|^2 + z^T z\} \\ &- \gamma^2\rho^2\{\|w+v\|^2 + (1+\alpha) z^T z + \alpha\|(x+v)_+ - x\|^2 - w^T w \\ &- 2[(x+v)_+ - x]^T w\} + \|\bar{x} - \tilde{x}\|^2.\end{aligned}$$

By $\alpha\rho^2 = \rho - \rho^2$ (from (13)),

$$\begin{aligned}\Theta &\ge \gamma^2\rho^2 w^T w + 2\gamma\rho[(x+v)_+ - x]^T v + 2\gamma^2\rho^2[(x+v)_+ - x]^T w \\ &- (2\gamma\rho - \gamma^2\rho^2)\|(x+v)_+ - x\|^2 - \gamma^2\rho^2\|w+v\|^2 + \|\bar{x} - \tilde{x}\|^2.\end{aligned} \tag{31}$$

Now we are ready to prove the second step $\Theta \ge 0$.

Proof of $\Theta \ge 0$. Without loss of generality we may assume that x, v and w are scalars. We distinguish the following cases:

(1) $(x+v)_+ = x+v$. In this case (31) yields

$$\Theta \geq \gamma^2\rho^2 w^2 + 2\gamma\rho v^2 + 2\gamma^2\rho^2 vw - 2\gamma\rho v^2 + \gamma^2\rho^2 v^2 - \gamma^2\rho^2(w+v)^2 = 0.$$

(2) $(x+v)_+ = 0 \neq x+v$.

(2.1) $(x+v)_+ = 0$ and $\bar{x} < 0 (\tilde{x} = 0)$: Note that in this case $\bar{x} = x + \gamma\rho(w+v) < 0$, i.e., $\|\bar{x} - \tilde{x}\|^2 = \|x + \gamma\rho(w+v)\|^2$, and (31) gives

$$\begin{aligned}\Theta &\geq \gamma^2\rho^2 w^2 - 2\gamma\rho xv - 2\gamma^2\rho^2 xw - (2\gamma\rho - \gamma^2\rho^2)x^2 \\ &\quad - \gamma^2\rho^2(w+v)^2 + x^2 + 2\gamma\rho x(w+v) + \gamma^2\rho^2(w+v)^2 \\ &= \gamma^2\rho^2 w^2 + 2\gamma\rho(1-\gamma\rho)xw + (1-\gamma\rho)^2 x^2 \\ &= [\gamma\rho w + (1-\gamma\rho)x]^2 \geq 0.\end{aligned}$$

(2.2) $(x+v)_+ = 0$ and $\bar{x} = \tilde{x} > 0$. Note that $(x+v)_+ = 0$ implies $x+v \leq 0$, and $\bar{x} > 0$ means $x + \gamma\rho(w+v) > 0$. So it follows that

$$0 < x + \gamma\rho(w+v) < x + \gamma\rho(w+v) - \gamma\rho(x+v) = \gamma\rho w + (1-\gamma\rho)x.$$

Then from (31) and the above relation we have

$$\begin{aligned}\Theta &\geq \gamma^2\rho^2 w^2 - 2\gamma\rho xv - 2\gamma^2\rho^2 xw - (2\gamma\rho - \gamma^2\rho^2)x^2 - \gamma^2\rho^2(w+v)^2 \\ &= [\gamma\rho w + (1-\gamma\rho)x]^2 - [x + \gamma\rho(w+v)]^2 \geq 0.\end{aligned}$$

Therefore in all cases $\Theta \geq 0$, and the proof of the theorem is complete.

Because Theorem 3 is true for any $u^* \in \Omega^*$ and $\rho_k \geq \frac{1}{1+\|A\|^2}$, in fact we have proved

$$\text{dist}^2(u^{k+1}, \Omega^*) \leq \text{dist}^2(u^k, \Omega^*) - \frac{\Delta_1(2-\Delta_2)}{1+\|A\|^2}\|e(u^k)\|^2, \tag{32}$$

where

$$\text{dist}(u, \Omega^*) = \inf\{\|u - u^*\| \mid u^* \in \Omega^*\}.$$

Especially, if $\Delta_1 = \Delta_2 = 1$, then we have

$$\text{dist}^2(u^{k+1}, \Omega^*) \leq \text{dist}^2(u^k, \Omega^*) - \frac{1}{1+\|A\|^2}\|e(u^k)\|^2. \tag{33}$$

The function $\|e(u)\|$ measures how much u fails to be in Ω^*. (32) states that, if $\|e(u^k)\|$ is not too small, then we gain a 'big' profit from an iteration; conversely, if we get a very small profit from an iteration, this implies that $\|e(u^k)\|$ is already very small and u^k is a 'sufficiently good' approximation of a $u^* \in \Omega^*$. In fact, from (32) it is possible to prove that, if Ω^* is nonempty, then the convergence speed of Algorithm 1 is linear.

5. An Extension—The Scaled Algorithm

In this section we will briefly describe how a scaled PC algorithm can be obtained by using a quite similar approach as described before. Let

$$G = \begin{pmatrix} G_x & \\ & G_y \end{pmatrix},$$

where G_x is an $n \times n$ diagonal matrix and G_y is a symmetric $m \times m$ matrix, and both are positive definite. Note that the value of the measure function ψ is not invariant under transformation of the form

$$\begin{aligned} M &\longrightarrow \tilde{M} = G^{-\frac{1}{2}} M G^{-\frac{1}{2}}, \\ q &\longrightarrow \tilde{q} = G^{-\frac{1}{2}} q, \\ u &\longrightarrow \xi = G^{\frac{1}{2}} u. \end{aligned}$$

The resulting LCP however with $\tilde{M}$ and $\tilde{q}$ is equivalent to the LCP(M, q). In particular, since our method is based on the function ψ, it is sensitive to such transformations of M. To find the best transformation G (preconditioning) is the subject of further research. Here we state the method for a given preconditioning matrix G. From (6) follows that u^* is a solution of the LCP if and only if it is a zero of the function

$$\tilde{e}(u, G) := u - P_\Omega[u - G^{-1}(Mu + q)]. \tag{6'}$$

Similarly we set

$$\tilde{\psi}(u, G) := \|\tilde{e}(u, G)\|_G^2 \tag{7'}$$

and

$$\tilde{\varphi}(u, G) := \tilde{e}(u, G)^T (Mu + q). \tag{8'}$$

Then we have

$$u \in \Omega^* \iff \tilde{\psi}(u, G) = 0$$

and

$$\tilde{\varphi}(u, G) \geq \tilde{\psi}(u, G) \qquad \text{for all} \quad u \in \Omega. \tag{9'}$$

We take

$$\tilde{g}(u, G) := M^T \tilde{e}(u, G) + (Mu + q) \tag{10'}$$

as the search direction and have the following assertion:

$$(u - u^*)^T \tilde{g}(u, G) \geq \tilde{\varphi}(u, G) \qquad \text{for all } u \in \Omega \text{ and } u^* \in \Omega^*. \tag{11'}$$

From the above relations, we have the following scaled PC algorithm:

PC Algorithm 2.

Step 0. Assume Δ_1 and Δ_2 are fixed constants satisfying $0 < \Delta_1 \leq \Delta_2 < 2$. Let $\varepsilon > 0$ and $u^0 \in \Omega$. Set $k := 0$.

Step 1. Calculate $\tilde{e}(u^k, G)$. If $\dfrac{\|\tilde{e}(u^k, G)\|_\infty}{\|q\|_\infty} \le \varepsilon$, stop.

Step 2. Calculate $\tilde{g}(u^k, G)$. Set

$$\alpha_k = \frac{\|G^{-1}M^T\tilde{e}(u^k, G)\|_G^2}{\|\tilde{e}(u^k, G)\|_G^2}, \tag{12'}$$

$$\rho_k = \frac{1}{1+\alpha_k} \tag{13'}$$

and take

$$\gamma_k \in [\Delta_1, \Delta_2]. \tag{14'}$$

Step 3. Set

$$\bar{u}^k = u^k - \gamma_k\rho_k G^{-1}\tilde{g}(u^k, G), \tag{15'}$$

$$u^{k+1} = P_\Omega[\bar{u}^k]. \tag{16'}$$

Set $k := k+1$ and go to Step 1.

For this scaled algorithm, we have the following result.

Theorem 4. *The sequence $\{u^k\}$ generated by PC Algorithm 2 for linear programs in standard form satisfies*

$$\|u^{k+1} - u^*\|_G^2 \le \|u^k - u^*\|_G^2 - \gamma_k(2-\gamma_k)\rho_k\|\tilde{e}(u^k, G)\|_G^2. \tag{20'}$$

This theorem is the analogue of Theorem 3 applied to the system

$$\xi - P_\Omega[\xi - (\tilde{M}\xi + \tilde{q})] = 0,$$

where $\xi = G^{\frac{1}{2}}u$, $\tilde{M} = G^{-\frac{1}{2}}MG^{-\frac{1}{2}}$ and $\tilde{q} = G^{-\frac{1}{2}}q$. Note that

$$\alpha = \frac{\|G^{-\frac{1}{2}}M^T\tilde{e}(u^k, G)\|^2}{\|G^{\frac{1}{2}}\tilde{e}(u^k, G)\|^2} \le \|G^{-\frac{1}{2}}MG^{-\frac{1}{2}}\|^2.$$

Because Theorem 4 is true for any $u^* \in \Omega^*$ and $\rho_k \ge \dfrac{1}{1+\|G^{-\frac{1}{2}}MG^{-\frac{1}{2}}\|^2}$, in fact we have proved

$$\operatorname{dist}_G^2(u^{k+1}, \Omega^*) \le \operatorname{dist}_G^2(u^k, \Omega^*) - \frac{\Delta_1(2-\Delta_2)}{1+\|G^{-\frac{1}{2}}MG^{-\frac{1}{2}}\|^2}\|\tilde{e}(u^k, G)\|_G^2, \tag{32'}$$

where

$$\operatorname{dist}_G(u, \Omega^*) = \inf\{\|u-u^*\|_G \mid u^* \in \Omega^*\}.$$

6. Numerical Experiments

In this section we compare the efficiency of our new algorithm with other similar algorithms—our prime algorithm in [5] and [6] and the extra gradient method by Korpelevich [8]. As a test problem we consider the transportation problem with random right hand side. The transportation problem is a linear program of the form

$$\begin{aligned} \min \ & \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij}, \\ \text{s.t.} \ & \sum_{j=1}^{n} x_{ij} = s_i, \quad i = 1,\cdots,m, \\ & \sum_{i=1}^{m} x_{ij} = d_j, \quad j = 1,\cdots,n, \\ & x_{ij} \geq 0,\ i = 1,\cdots,m,\ j = 1,\cdots,n. \end{aligned}$$

where

$m=$ number of sources, $n=$ number of destinations,

$s_i=$ supply at source i, $d_j=$ demand at destination j.

For feasibility, it is necessary that we have

$$\sum_{i=1}^{m} s_i = \sum_{j=1}^{n} d_j.$$

We generate random test transportation problems as follows: we take

$$\begin{aligned} s_i &= 50 \times \operatorname{ran}(*) + 50 \quad \text{for } i = 1,\cdots,m, \\ \tilde{d}_j &= 50 \times \operatorname{ran}(*) + 50 \quad \text{for } j = 1,\cdots,n, \end{aligned}$$

and set

$$d_j = t\tilde{d}_j, \qquad j = 1,\cdots,n,$$

where ran $(*)$ denotes a random variable in $(0,1)$ and

$$t := \sum_{i=1}^{m} s_i / \sum_{j=1}^{n} \tilde{d}_j.$$

Then, the system is balanced in the sense that the total supply equals the total demand. Further, we choose the components c_{ij} of the vector c randomly in (0,100).

All codes were written in FORTRAN and run on a VAX 8810 computer of the Computing Center of the University of Würzburg. The calculations were started with $u^0 := 0$, and stopped as soon as

$$\max\left\{\frac{\|Ax-b\|_\infty}{\|b\|_\infty}, \frac{\|x-[x+(A^Ty-c)]_+\|_\infty}{\|c\|_\infty}\right\} \leq \varepsilon$$

for some $\varepsilon > 0$. The termination criterion was checked at every tenth iteration. The CPU–times refer only to the runs with the highest accuracy $\varepsilon = 10^{-3}$.

6.1. The prime PC algorithm

The iterative scheme of the prime PC algorithm in [5] and [6] can be written as

$$u^{k+1} = P_\Omega[u^k - \rho_k g(u^k)]$$

where ρ_k is calculated in each iteration by (17). The numerical results with prime PC algorithm are given in Table 1.

Table 1. Prime PC Algorithm[6]

# orig.	# dest.	# var.	# iteration for $\varepsilon =$			CPU
m	n	mn	0.1	0.01	0.001	sec
40	50	2000	30	100	510	15.80
75	80	6000	20	180	810	79.88
80	125	10000	50	280	740	125.17

6.2. Korpelevich's method

As pointed in [7], in the case of solving linear programs, the PC method with constant step-size (ρ_k is given by (18)) can be viewed as an extension of the extra gradient method proposed by Korpelevich [8]. Korpelevich's iterative scheme is the following

$$\hat{u}^k = P_\Omega[u^k - \lambda(Mu^k + q)], \quad u^{k+1} = P_\Omega[u^k - \lambda(M\hat{u}^k + q)].$$

It is proved (in [8]) that for all $0 < \lambda < 1/\|M\|$, the sequence $\{u^k\}$ satisfies

$$\|u^{k+1} - u^*\|^2 \le \|u^k - u^*\|^2 - (1 - \lambda^2\|M\|^2)\|u^k - \hat{u}^k\|^2$$

and it is linearly convergent. Note that for linear programming, $\|M\| = \|A\|$. The constraint matrix of a transportation problem with m origins and n destinations is an $(m+n) \times mn$ matrix of the form

$$A = \begin{pmatrix} 1 & \cdots & 1 & & & & & & & & \\ & & & 1 & \cdots & 1 & & & & & \\ & & & & & & \cdots & & & & \\ & & & & & & & & 1 & \cdots & 1 \\ 1 & & & 1 & & & & & 1 & & \\ & \ddots & & & \ddots & & \cdots & & & \ddots & \\ & & 1 & & & 1 & & & & & 1 \end{pmatrix}.$$

Because

$$AA^T = \begin{pmatrix} n & & & & & 1 & \cdot & \cdot & \cdot & 1 \\ & \cdot & & & & \cdot & \cdot & \cdot & \cdot & \cdot \\ & & \cdot & & & \cdot & \cdot & \cdot & \cdot & \cdot \\ & & & \cdot & & \cdot & \cdot & \cdot & \cdot & \cdot \\ & & & & n & 1 & \cdot & \cdot & \cdot & 1 \\ 1 & \cdot & \cdot & \cdot & 1 & m & & & & \\ \cdot & \cdot & \cdot & \cdot & \cdot & & \cdot & & & \\ \cdot & \cdot & \cdot & \cdot & \cdot & & & \cdot & & \\ \cdot & \cdot & \cdot & \cdot & \cdot & & & & \cdot & \\ 1 & \cdot & \cdot & \cdot & 1 & & & & & m \end{pmatrix},$$

it is easy to see that

$$\min\{m,n\} \le \lambda_{max}(AA^T) = \|A\|^2 \le 2 \cdot \max\{m,n\}.$$

We take $\tau := \frac{1}{2}(\sqrt{\min\{m,n\}} + \sqrt{2 \cdot \max\{m,n\}})$ as an estimate for $\|A\|$ and as in [8] we choose $\lambda = \dfrac{1}{\sqrt{2}\tau}$ as the constant step-size. The numerical results with Korpelevich's method are in Table 2.

Table 2. Korpelevich's Algorithm [8]

# orig.	# dest.	# var.	# iteration for $\varepsilon =$			CPU
m	n	mn	0.1	0.01	0.001	sec
40	50	2000	50	410	2740	69.27
75	80	6000	60	570	2930	231.69
80	125	10000	60	750	3390	444.02

6.3. Our new PC algorithm–PC Algorithm 1 The iterative scheme of our PC algorithm 1

$$u^{k+1} = P_\Omega[u^k - \gamma_k \rho_k g(u^k)]$$

where ρ_k is calculated in each iteration by (13) and $0 < \Delta_1 \le \gamma_k \le \Delta_2 < 2$. In our test problems we take $\gamma_k \equiv 1.95$ and the numerical results with this algorithm are given in Table 3.

Table 3. The new PC algorithm

# orig.	# dest.	# var.	# iteration for $\varepsilon =$			CPU
m	n	mn	0.1	0.01	0.001	sec
40	50	2000	20	90	400	12.38
75	80	6000	20	160	650	62.69
80	125	10000	40	150	530	89.61

Because in practical problems the data of b and c come from experiments or from statistics, a threshold $\varepsilon = 0.001$ is sufficient for most practical applications. Our numerical results show that both our prime PC algorithm and new PC algorithm converge much faster than Korpelevich's extra gradient method. For $\varepsilon = 0.001$, our PC methods require 70-80 percent fewer iterations than Korpelevich's method. In general, our new PC algorithm is more efficient than the PC algorithm with constant step-size; the reason is that $\rho_{\text{new}} \geq \rho_{\text{const}}$ (see (19)). Comparing Table 3 with Table 1, we conclude that the new PC algorithm also works better than the prime one at least in some cases.

7. Conclusion

For fixed precision, the number of iterations of our PC algorithms is rather insensitive to the increase of the size of the problem (number of variables). In view of the moderate number of iterations and the low cost of each iteration, the PC method seems to be an efficient method for solving large sparse linear programming problems. Moreover, since the PC method is easy to parallelize, it may be even more favorable for parallel computation.

We point out however that the methods are convergent only if $\Omega^* \neq \emptyset$ and the convergence results of our new PC algorithms are proved only for linear programming in standard form. If $\Omega^* = \emptyset$, we conjecture that the iterates diverge to infinity.

Acknowledgement. Thanks are due to F. Jarre for his comments on this paper.

References

[1] E. Blum and W. Oettli, Mathematische Optimierung, Econometrics and Operations Research XX, Springer-Verlag, Berlin, Heidelberg, New York, 1975.

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

[3] V. F. Demyanov and A. B. Pevnyi, Numerical methods for finding saddle points, *USSR Comput. Math. Math. Phys.*, **12** (1972), 11-52.

[4] L. G. Gubin, B. T. Polyak and E. V. Raik, The method of projections for finding the common point of convex sets, *USSR Comput. Math. Math. Phys.*, **7** (1967), 1-24.

[5] B. S. He, A saddle point algorithm for linear programming, *Nanjing Daxue Xuebao, Shuxue Banniankan*, **6** (1989), 41-48.

[6] B, S. He, A projection and contraction method for a class of linear complementarity problem and its application in convex quadratic programming, *Applied Mathematics and Optimization*, **25** (1992), 247–262.

[7] B. S. He, On a class of iterative projection and contraction methods for linear programming, *JOTA*, **78** (1993).

[8] G. M. Korpelevich, The extragradient method for finding saddle points and other problems, *Ekonomika i matematicheskie metody*, **12** (1976), 747–756.

[9] O. L. Mangasarian, Solution of symmetric linear complementarity problems by iterative methods, *JOTA*, **22** (1979), 465–485.

[10] H. Uzawa, Iterative methods for concave programming, in: Studies in Linear and Nonlinear Programming (K. J. Arrow, L. Hurwicz and H. Uzawa, ed.), Stanford University Press, Stanford, 1958.