Speeding up Chubanov's Basic Procedure

Kees Roos, Delft University of Technology, c.roos@tudelft.nl

September 18, 2014

Abstract

It is shown that a recently proposed method by Chubanov for solving linear homogeneous systems with positive variables can be improved, both theoretically and computationally.

Keywords: linear homogeneous systems, algorithm, polynomial-time

Contents

1 Introduction **1**

2 Preliminaries **2**

3 More on cut-generating vectors **5**

4 The Basic Procedure **6**

5 Exploiting cuts **10**

6 Computational results **10**

7 Concluding remarks **13**

1 Introduction

Recently Chubanov [1, 2] presented a new algorithm that finds in polynomial time a solution of the system

$$Ax = 0,\ x > 0, \tag{1}$$

or establishes that no such solution exists. Here A is an $m \times n$ matrix with $m < n$, and $\operatorname{rank}(A) = m$. In the algorithm the author uses a nonzero vector $y \geq 0$ that is updated in each iteration and eventually serves to decide which of the two cases occurs. If a solution exists, then the algorithm generates a solution and if no solution exists then it provides a certificate for infeasibility of (1).

In hindsight, duality theory helps to understand the role of the vector y in Chubanov's approach. By Stiemke's lemma [11], which is a variant of Farkas's lemma, (1) has no solution if and only if the system

$$A^T u \geq 0, \quad A^T u \neq 0 \tag{2}$$

has a solution. Now one has $y = A^T u$ for some u if and only if $P_A y = 0$, where P_A denotes the orthogonal projection onto the null space of A. It follows that system (2) has a solution if and only if the system

$$P_A y = 0, \quad y \geq 0, \quad y \neq 0 \tag{3}$$

has a solution. Chubanov's algorithm can be viewed as a systematic method for searching a vector y that satisfies (3). This search occurs in his so-called Basic Procedure (BP). In each iteration the BP constructs a new vector y such that $\|P_A y\|$ strictly decreases. In this paper we fucus on this procedure and we show that its performance can be improved significantly.

If the BP generates a vector y that does not yet make clear what the status is of (1) then Chubanov showed that it gives rise to a cut, i.e., an inequality $x_k \leq \frac{1}{2}$ for some k that is satisfied by all feasible solutions x such that $x \leq e$. If this happens we call y a *cut-generating vector*, or shortly *cutting vector*.

The contribution of this paper is twofold:

- we modify the stopping criterium for the BP, thereby improving the theoretical iteration bound by a factor n;
- we modify the construction of the new vector y such that $\|P_A y\|$ decreases faster.

These claims are justified both theoretically, and computationally.

The outline of the paper is as follows. In the next section we introduce some notations and terminology. Lemma 2.1 in this section is new; as we make clear it gives in some sense the tightest possible cuts that can be obtained from a given vector y. In Section 3 we discuss the cuts used in [1, 2] and we prove that the cuts in Lemma 2.1 are at least as tight. Our BP is presented in Section 4. It has the same structure as Chubanov's BP but differs in the way the vector y is constructed. Lemma 4.1 provides theoretical evidence that $\|P_A y\|$ will decrease faster. Also the stopping criterium for the BP is different. It uses the cuts of Lemma 2.1 and we show in Theorem 4.4 that it reduces the iteration bound by a factor n. In Section 5 we briefly indicate how cutting vectors are used in Chubanov's Main Algorithm. Section 6 presents computational results that further support the above claims, and Section 7 contains some concluding remarks.

2 Preliminaries

Since problem (1) is homogeneous, it may be assumed that $0 < x \leq e$, where e denotes the all-one vector of length n. Hence, without loss of generality we may equally well consider the problem

$$Ax = 0, \; x \in (0, 1]^n. \tag{4}$$

Let $\mathcal{N}_A$ denote the null space of the $m \times n$ matrix A and $\mathcal{R}_A$ its row space. So

$$\mathcal{N}_A := \{x \in \mathbf{R}^n \;:\; Ax = 0\}, \quad \mathcal{R}_A := \left\{A^T u \;:\; u \in \mathbf{R}^m\right\}.$$

We denote the orthogonal projections of $\mathbf{R}^n$ onto $\mathcal{N}_A$ and $\mathcal{R}_A$ as P_A and Q_A respectively:

$$P_A := I - A^T \left(AA^T\right)^{-1} A, \quad Q_A := A^T \left(AA^T\right)^{-1} A.$$

Note that our assumption rank $(A) = m$ implies that the inverse of AA^T exists. Obviously we have

$$I = P_A + Q_A, \quad P_A Q_A = 0, \quad AP_A = 0, \quad AQ_A = A.$$

Now let $y \in \mathbf{R}^n$, $0 \neq y \geq 0$. In the sequel we assume without loss of generality that $e^T y = 1$. We also use the notation

$$z = P_A y, \quad v = Q_A y.$$

So z and v are the orthogonal components of y in the spaces $\mathcal{N}_A$ and $\mathcal{R}_A$ respectively:

$$y = z + v, \quad z \in \mathcal{N}_A, \quad v \in \mathcal{R}_A.$$

These vectors play an crucial role in the approach of Chubanov. This is due to the fact that if $z > 0$ then z solves the *primal problem* (1), because $z \in \mathcal{N}_A$, and if $z = 0$ then $v = y$ solves the *dual problem* (3), because $v \in \mathcal{R}_A$.

The BP generates (in strongly polynomial time) a vector y such that one of the following three cases occurs:

I. $z = P_A y$ is feasible for (1);

II. $z = 0$, meaning that y satisfies (3);

III. there exists an index k such that v satisfies

$$\sum_{i=1}^{n} \left[\frac{-v_i}{v_k}\right]^+ \leq \frac{1}{2}, \tag{5}$$

where $[a]^+$ arises from a by replacing its negative entries by zero.

In the first two cases the status of (1) is clear: in case I we have a solution of (1), and in case II a certificate for its infeasibility. It will be convenient to abuse language and call the vector y *primal feasible* and *dual feasible*, respectively, in these cases.

In the third case y is a cutting vector. The reason for this name is that if (5) holds then we have $x_k \leq \frac{1}{2}$ for all solutions of (22), thus cutting off halve of the feasible region of (4). This is a consequence of the next lemma.

Lemma 2.1 *Let x be feasible for (4). Then every nonzero element v_k of v gives rise to a cut, according to*

$$x_k \leq \sum_{i=1}^{n} \left[\frac{-v_i}{v_k}\right]^+. \tag{6}$$

Proof: Since $x \in \mathcal{N}_A$ and $v \in \mathcal{R}_A$ we have $v^T x = 0$. Now suppose $v_k < 0$. Then also using $0 \leq x \leq e$ we obtain

$$-v_k x_k = \sum_{i \neq k} v_i x_i \leq \sum_{i,\, v_i > 0} v_i x_i \leq \sum_{i,\, v_i > 0} v_i,$$

On the other hand, if $v_k > 0$ we obtain in the same way

$$v_k x_k = -\sum_{i \neq k} v_i x_i \leq \sum_{i,\, v_i < 0} -v_i x_i \leq \sum_{i,\, v_i < 0} -v_i.$$

These two inequalities imply the inequality in the lemma, as one easily verifies. $\square$

Remark 2.2 *It cannot happen that $v \neq 0$ and all nonzero entries of v have the same sign, unless (4) is infeasible. This follows from (6), which gives $x_k = 0$ for every k with $v_k \neq 0$.*

We call the inequality (6) nonvoid if it the upper bound is less than 1. It may easily be verified that this holds if and only if one of the following two inequalities is satisfied:

$$v_k > -\sum_{v_i < 0} v_i \quad \text{if} \quad v_k > 0 \tag{7}$$

$$v_k < -\sum_{v_i > 0} v_i \quad \text{if} \quad v_k < 0. \tag{8}$$

One also easily verifies that (7) implies $e^T v > 0$ and (8) implies $e^T v < 0$. This makes clear that if y is a cutting vector then $e^T v \neq 0$. Moreover, if $e^T v > 0$ then only positive elements of v may give rise to a cut and if $e^T v < 0$ then only negative elements of v may give rise to a cut. Also note that $e^T y = 1$ implies $\|y\| \leq 1$, which gives $\|v\| \leq 1$.

Before proceeding we present a simple example, where we use that the upper bound for x_k in (6) is homogeneous in v.

Example 2.3 *By way of example we consider the case where v (up to a factor 10) is given by*

$$v = \begin{bmatrix} 3 \\ 4 \\ -2 \\ 0 \\ 2 \\ 6 \end{bmatrix}.$$

Since $e^T v = 13 > 0$ only positive entries v_k in v may give rise to a nonvoid cut, and this happens if v_k exceeds

$$-\sum_{v_i < 0} v_i = 2.$$

Thus we obtain the following nonvoid cuts:

$$x_1 \leq \frac{2}{3}, \quad x_2 \leq \frac{1}{2}, \quad x_6 \leq \frac{1}{3}.$$

Before dealing with the procedure that generates a vector y satisfying I, II or III we include a section that discusses two other methods that produce a cutting vector, at the same time showing that these cuts are less tight than (6).

3 More on cut-generating vectors

In the previous section it became clear that a vector y may give rise to a cut for problem (4). In [1, 2] Chubanov worked with another cut, namely

$$x_k \leq \frac{\sqrt{n}\,\|z\|}{y_k}. \tag{9}$$

Of course, this inequality makes sense only if the last expression is smaller than 1, because a priori it is known that $x_k \leq 1$, for each k. If this expression is not larger than $\frac{1}{2}$ we obtain $x_k \leq \frac{1}{2}$.

In [2] Chubanov mentioned that also other cuts can be obtained from y by using the Duality Theorem for Linear Optimization. In this section we briefly discuss this method and we show that a little more sophisticated use of this theorem leads to the new cuts in Lemma 2.1. It also follows from the analysis in this section that the new cuts are the strongest.

Fixing k, Chubanov [2] considers the LO-problem

$$\max\left\{x_k \;:\; Ax = 0,\, x \in [0,1]^n\right\}.$$

The dual problem is

$$\min\left\{e^T w \;:\; A^T\xi + w \geq e_k,\, w \geq 0\right\} = \min\left\{e^T\left[e_k - u\right]^+ \;:\; P_A u = 0\right\}.$$

The above equality uses that $u = A^T\xi$ for some ξ if and only if $P_A u = 0$. Hence, if $y_k > 0$ we may take $u = \frac{v}{y_k}$, with v as in the previous section. It then follows from the Duality Theorem for Linear Optimization that

$$x_k \leq e^T\left[e_k - \frac{v}{y_k}\right]^+. \tag{10}$$

One has

$$e^T\left[e_k - \frac{v}{y_k}\right]^+ \leq e^T\left[\frac{z}{y_k}\right]^+ = \frac{e^T z^+}{y_k} \leq \frac{\sqrt{n}\|z^+\|}{y_k} \leq \frac{\sqrt{n}\,\|z\|}{y_k},$$

showing that cut (9), which was used in [1, 2], is implied by (10).

Yet we present another way to obtain the cuts in Lemma 2.1, thereby showing that these cuts are tighter than (10) and hence also than (9). Instead of $u = \frac{v}{y_k}$ we put more generally $u = \alpha v$, with $\alpha \in \mathbf{R}$. We then have $x_k \leq q(\alpha)$ for every α, where the function $q(\alpha)$ is defined by

$$q(\alpha) := e^T\left[e_k - \alpha v\right]^+ = \left[1 - \alpha v_k\right]^+ + \sum_{i \neq k}\left[-\alpha v_i\right]^+, \quad \alpha \in \mathbf{R}.$$

One may easily verify that $q(\alpha)$ is a nonnegative piecewise linear convex function with a breakpoint at $\alpha = 0$ and, if $v_k \neq 0$, another breakpoint at $\alpha = \frac{1}{v_k}$. Since $q(\alpha)$ is convex it attains its minimal value at a breakpoint. The breakpoint at $\alpha = 0$ yields the void inequality $x_k \leq q(0) = 1$. So only the breakpoint at $\alpha = \frac{1}{v_k}$ is of interest, and this yields exactly the inequality in Lemma 2.1 (because the first term in the expression for $q(\alpha)$ vanishes at this breakpoint).

We conclude from the above analysis that for each nonzero $y \geq 0$ and for each k one has

$$\min\left(1, \sum_{i=1}^{n}\left[\frac{-v_i}{v_k}\right]^+\right) \leq e^T\left[e_k - \frac{v}{y_k}\right]^+ \leq \frac{\sqrt{n}\,\|z\|}{y_k}. \tag{11}$$

Since the left expression represents the minimal value of $q(\alpha)$, each of the three expressions is an upper bound for x_k.

It may be worth noting that the expression in the middle equals $q(\frac{1}{y_k})$. Hence it is equal to the new bound if and only if $y_k = v_k$, i.e., if and only if $z_k = 0$. Moreover, it yields a nonvoid upper bound only if $v_k > 0$. This easily follows because if $v_k \leq 0$ then the value of the k-th term alone already is at least 1.

Finally, in the BP we need to compute these expressions only for the index k that gives the smallest value. For the expression at the left this is the (or an) index such that $v_k = \max(v)$, and for the other two expressions the (or an) index such that $y_k = \max(y)$. In each case this computation requires $O(n)$ time.

4 The Basic Procedure

If y is such that the status of (1) is not yet decided then $z \neq 0$ and $z_k \leq 0$ for at least one index k. Hence we may find a nonempty set K of indices such that

$$\sum_{k\in K} z_k \leq 0.$$

Denoting the k-th column of P_A as p^k, we have $p^k = P_A e_k$, where e_k denotes the k-th unit vector. We define

$$e_K := \frac{1}{|K|}\sum_{k\in K} e_k, \quad p_K := P_A e_K = \frac{1}{|K|}\sum_{k\in K} p^k. \tag{12}$$

Note that $0 \neq e_K \geq 0$, and $e^T e_K = 1$. If $p_K = 0$ ($p_K > 0$), then e_K is dual (primal) feasible and we are done. Hence, we may assume that $p_K \neq 0$. Since P_A is a projection matrix we have $P_A^2 = P_A$. Hence we obtain $P_A z = P_A^2 y = P_A y = z$. This implies $z^T p^k = z^T P_A e_k = z^T e_k = z_k$ for each k. Thus we obtain

$$z^T p_K = \frac{1}{|K|}\sum_{k\in K} z^T p^k = \frac{1}{|K|}\sum_{k\in K} z_k \leq 0.$$

Since $z \neq 0$, $p_K \neq 0$ and $z^T p_K \leq 0$, in the equation

$$\|z - p_K\|^2 = \left(\|z\|^2 - z^T p_K\right) + \left(\|p_K\|^2 - z^T p_K\right) \tag{13}$$

the two bracketed terms are both positive. Therefore, we may define a new y-vector, denoted by $\tilde{y}$, according to

$$\tilde{y} = \alpha y + (1-\alpha) e_K, \quad \alpha = \frac{\|p_K\|^2 - z^T p_K}{\|z - p_K\|^2} = \frac{p_K^T(p_K - z)}{\|z - p_K\|^2}. \tag{14}$$

Because of (13), α is well-defined and $\alpha \in (0,1)$. Since $y \geq 0$ and $e_K \geq 0$, we may conclude that $\tilde{y} \geq 0$ and, since $e^T y = e^T e_K = 1$, also $e^T \tilde{y} = 1$.

Algorithm 1: $[y, z, \text{case}]$ =BASIC PROCEDURE(P_A, y)

```
1:  INITIALIZE: $z = P_A y$; case $= 0$
2:  while bound $(y) > \frac{1}{2}$ and case $= 0$ do
3:      if $z > 0$ then
4:          case $= 1$   ($y$ is primal feasible); return
5:      else
6:          if $z = 0$ then
7:              case $= 2$  ($y$ is dual feasible); return
8:          else
9:              find $K \neq \emptyset$ such that $\sum_{k \in K} z_k \leq 0$
10:             $\alpha = p_K^T(p_K - z)/\|z - p_K\|^2$
11:             $y = \alpha y + (1 - \alpha)e_K$
12:             $z = \alpha z + (1 - \alpha)p_K$ $(= P_A y)$
```

The transformation (14) from y to $\tilde{y}$ is the key element in Algorithm 1. It iterates (14) until y is primal feasible or dual feasible or a cutting vector. In this algorithm bound (y) denotes one of the three expressions in (11).

The only two differences from the BP of Chubanov are that he always takes a singleton for K and for bound (y) he uses (9). We provide theoretical evidence in Lemma 4.1 that the use of larger sets K may improve the performance of the Basic Procedure; as will become clear in Section 6, in our computations the improvement is considerable. However, it may deteriorate the theoretical performance of the BP, because the computation of p_K requires $O(n\,|K|)$ time instead of $O(n)$. A remedy for this is to limit the size of K to $|K| = O(1)$, then the order of the time bound per iteration of the BP does not change!

Computationally, the use of the cut in Lemma 2.1 turns out to have a similar effect: it reduces the computational time of the BP notably. This is supported by Theorem 4.4 below, which implies that it reduces the time complexity of the BP with a factor n.

The following lemma generalizes [1, Lemma 2.1] and [2, Lemma 2.1], where it was assumed that K is a singleton.

Lemma 4.1 *Let $z \neq 0$ and $p_K \neq 0$. With $\tilde{z} := P_A\tilde{y}$, one has*

$$\frac{1}{\|\tilde{z}\|^2} \geq \frac{1}{\|z\|^2} + |K|\,. \tag{15}$$

Proof: We have

$$\tilde{z} = \alpha P_A y + (1 - \alpha)P_A e_K = \alpha z + (1 - \alpha)p_K = p_K + \alpha(z - p_K).$$

Hence,

$$\|\tilde{z}\|^2 = \alpha^2\|z - p_K\|^2 + 2\alpha p_K^T(z - p_K) + \|p_K\|^2.$$

The value of α that minimizes this expression is precisely as given in (14). It follows that

$$\|\tilde{z}\|^2 = \|p_K\|^2 - \frac{\left[p_K^T(z - p_K)\right]^2}{\|z - p_K\|^2} = \frac{\|p_K\|^2\|z\|^2 - (z^Tp_K)^2}{\|p_K\|^2 + \|z\|^2 - 2z^Tp_K} \leq \frac{\|p_K\|^2\|z\|^2}{\|z\|^2 + \|p_K\|^2},$$

where we used again that $z^T p_K \leq 0$. Since P_A is a projection matrix, $\|P_A e_K\| \leq \|e_K\|$. So we may write

$$\|p_K\|^2 = \|P_A e_K\|^2 \leq \|e_K\|^2 = \left\| \frac{1}{|K|} \sum_{k\in K} e_k \right\|^2 = \frac{1}{|K|^2} \left\| \sum_{k\in K} e_k \right\|^2 = \frac{|K|}{|K|^2} = \frac{1}{|K|}.$$

It follows that

$$\frac{1}{\|\tilde{z}\|^2} \geq \frac{1}{\|z\|^2} + \frac{1}{\|p_K\|^2} \geq \frac{1}{\|z\|^2} + |K|, \tag{16}$$

as desired. □

The next theorem is due to Chubanov. It provides an iteration bound for the BP when using for bound (y) the third expression in (9). We include its proof, as a preparation for the case where the BP uses for bound (y) the first expression in (9).

Theorem 4.2 *When using the cuts in (9) the BP stops after at most $4n^3$ iterations.*

Proof: As before, we assume that $e^T y = 1$, $y \geq 0$ and $z = P_A y$. If y is cutting then the BP requires only 1 iteration. Otherwise y we have

$$\frac{\sqrt{n}\|z\|}{\max(y)} > \frac{1}{2}.$$

Since $\max(y) \geq 1/n$, this implies

$$\frac{1}{\|z\|^2} < 4n^3. \tag{17}$$

If during the execution of Algorithm 1 it happens that $z > 0$ or $z = 0$ then the BP immediately stops. Otherwise, since $|K| \geq 1$, each iteration of the while loop increases $1/\|z\|^2$ by at least 1. Hence, after at most $4n^3$ executions of the while loop the algorithm yields a vector y that is primal feasible (case = 1) or dual feasible (case = 2) or such that $1/\|z\|^2 \geq 4n^3$. In the last case y is cutting (case = 0). □

Since each execution of the while loop requires at most $O(n)$ time, the BP will require at most $O(n^4)$ time. It must be mentioned that for solving (1) one needs to call the BP several times. This happens by Chubanov's Main Algorithm (see Section 5). Surprisingly, Chubanov was able to show that on average the BP then will require only $O(n^3)$ time.

Below we improve this result by using for bound (y) the first expression in (9). As we will see then $O(n^3)$ becomes a worst-case bound. As in the proof of Theorem 4.2, we start by deriving a lower bound for $\|z\|$. This is now more cumbersome and requires a separate lemma. In this lemma we use a parameter τ; for our current goal one must put $\tau = 2$.

Lemma 4.3 *Let $\tau \geq 1$ and (y, z, v) a triple of vectors in $\mathbf{R}^n$ such that*

$$y \geq 0,\ e^T y = 1,\ y = z + v,\ z^T v = 0,\ v \neq 0,$$

and

$$\sum_{i=1}^{n} \left[\frac{-v_i}{v_k}\right]^+ \geq \frac{1}{\tau}, \quad \forall k \text{ such that } v_k \neq 0. \tag{18}$$

Then

$$\|z\|^2 \geq \frac{1}{(n-1)(1+(n-1)\tau^2)}.$$

Proof: One easily verifies that v satisfies (18) if and only if

$$\sum_{v_i<0} -v_i \geq \frac{1}{\tau}\max(v), \tag{19}$$

$$\sum_{v_i>0} v_i \geq -\frac{1}{\tau}\min(v). \tag{20}$$

From (19) we deduce that if v has a positive entry, it must also have negative entries, and from (20) that if v has a negative entry, there must also be negative entries. So v has both positive and negative entries. Without loss of generality we assume that $v_1 \leq v_2 \leq \ldots \leq v_n$. We then have $v_1 < 0$ and $v_n > 0$.

Since $z = y - v$ and $z^T v = 0$ we have $y^T v = v^T v = \|v\|^2$. Also using $\|y\|^2 = \|z\|^2 + \|v\|^2$ we get

$$\|z\|^2 = \|y\|^2 - y^T v.$$

To prove the lemma we should find the smallest possible value of $\|z\|$. For this value we must have $y^T v > 0$, and, moreover, if $v_i \leq 0$ then $y_i = 0$. Defining $\bar{y} = y_{2:n}$ and $\bar{v} = v_{2:n}$ and taking $y_1 = 0$, because $v_1 < 0$, we obtain

$$\|z\|^2 = \|\bar{y}\|^2 - \bar{y}^T \bar{v}.$$

Given $\|\bar{y}\|$ and $\|\bar{v}\|$ this expression is minimal if $\bar{y}^T\bar{v}$ is maximal, i.e., if $\bar{v} = \lambda\bar{y}$ for some positive λ. Then $\bar{z} = \bar{y} - \bar{v} = (1-\lambda)\bar{y}$. Because $z^T v = 0$ and $z_1 = y_1 - v_1 = -v_1$ we obtain

$$0 = -v_1^2 + (1-\lambda)\bar{y}^T\bar{v} = -v_1^2 + \frac{1-\lambda}{\lambda}\bar{v}^T\bar{v} = -v_1^2 + \frac{1-\lambda}{\lambda}\|\bar{v}\|^2,$$

which implies

$$\lambda = \frac{\|\bar{v}\|^2}{v_1^2 + \|\bar{v}\|^2}. \tag{21}$$

Hence $\lambda < 1$ and

$$\|z\|^2 \geq \|\bar{y}\|^2 - \lambda\bar{y}^T\bar{y} = (1-\lambda)\|\bar{y}\|^2 = \frac{v_1^2}{v_1^2 + \|\bar{v}\|^2}\|\bar{y}\|^2.$$

The last expression is increasing in $\|\bar{y}\|^2$ and decreasing in $\|\bar{v}\|$. Since $y_1 = 0$ and $e^T y = 1$ we have $\|\bar{y}\|^2 \geq 1/(n-1)$. On the other hand, $\|\bar{v}\|$ is maximal if all entries of $\bar{v}$ are equal to $-\tau v_1$. This follows since all entries of $\bar{v}$ are equal and positive, and by (19) bounded from above by $-\tau v_1$. Thus we obtain $\|\bar{v}\|^2 \leq (n-1)\tau^2 v_1^2$. These bounds on $\|\bar{y}\|^2$ and $\|\bar{v}\|^2$ imply that the smallest possible value of $\|z\|$ satisfies

$$\|z\|^2 \geq \frac{v_1^2}{v_1^2 + (n-1)\tau^2 v_1^2}\frac{1}{n-1} = \frac{1}{(n-1)(1+(n-1)\tau^2)},$$

proving the lemma. □

Theorem 4.4 *When using the cuts in (6) the BP stops after at most* $(n-1)(4n-3)$ *iterations.*

Proof: Using Lemma 4.3, with $\tau = 2$, the proof goes in the same way as the proof of Theorem 4.2. $\square$

Obviously the iteration bound – and hence also the time complexity of the BP – is now a factor n better than in Theorem 4.2. Before presenting some computational results we briefly indicate how the above results can be used to solve problem (1) in polynomial time. This is the subject of the next section.

5 Exploiting cuts

For completeness' sake we consider it useful to discuss shortly the ideas underlying Chubanov's Main Algorithm (MA), thereby also showing why it is a polynomial-time method for solving (1).

Chubanov's MA maintains a vector $d > 0$ such that $x \leq d \leq e$ holds for every feasible solution x of (4). Note that x is feasible for (4) if and only if $x' = D^{-1}x$ is feasible for

$$ADx = 0, \;\; x \in (0, 1]^n, \tag{22}$$

where $D = \operatorname{diag}(d)$. This problem is of the same type as problem (4), since it arises from (4) simply by replacing A by AD.

Initially one has $d = e$, but each time that the BP generates a cutting vector y one of the entries in d is divided by 2. Then the MA calls the BP again with P_{AD} as input, and so on.

There are two ways to decide from d that (1) is infeasible. One way uses a well known result of Khachiyan, namely that there exists a positive number τ such that $1/\tau = O(2^{\text{size}(A)})$ with the property that the positive coordinates of the basic feasible solutions of (4) are bounded from below by τ. Here size(A) denotes the binary size of matrix A [7, 9, 10]. Hence, if one of the entries of d becomes less that τ then we may conclude that (1) is infeasible. From this one easily derives that the number of calls of the BP by the MA is at most $O(n \cdot \text{size}(A))$. The main computational task of the MA is the computation of P_{AD}. This can be done in $O(n^3)$ time [4, 9], and even faster with the Sherman-Morrisen-Woodbury formula [6]. Provided that $O(|K|) = 1$, the time complexity of the BP is also $O(n^3)$. Hence, in total the algorithm will require $O(n^4 \cdot \text{size}(A))$ time. This result was first obtained in [1, 2], but in a much more cumbersome way.

A second way to derive from d that (1) is infeasible is the observation that if (4) is feasible then it has a solution with $x_i = 1$ for at least one index i. This implies that if during the course of the MA we find that $d < e$ then (1) must be infeasible. Though this observation, which seems to be new, does not improve the above iteration bound, it might be interesting from a computational point of view.

6 Computational results

We first illustrate the effect of allowing the set K to be larger than a singleton. For this we refer to Figures 1 and 2. These figures were obtained by applying Algorithm 1 to a randomly generated matrix A of size 25×50. In that case the algorithm needs 256 iterations if we take for K the singleton k for which z_k is minimal. On the other hand, if

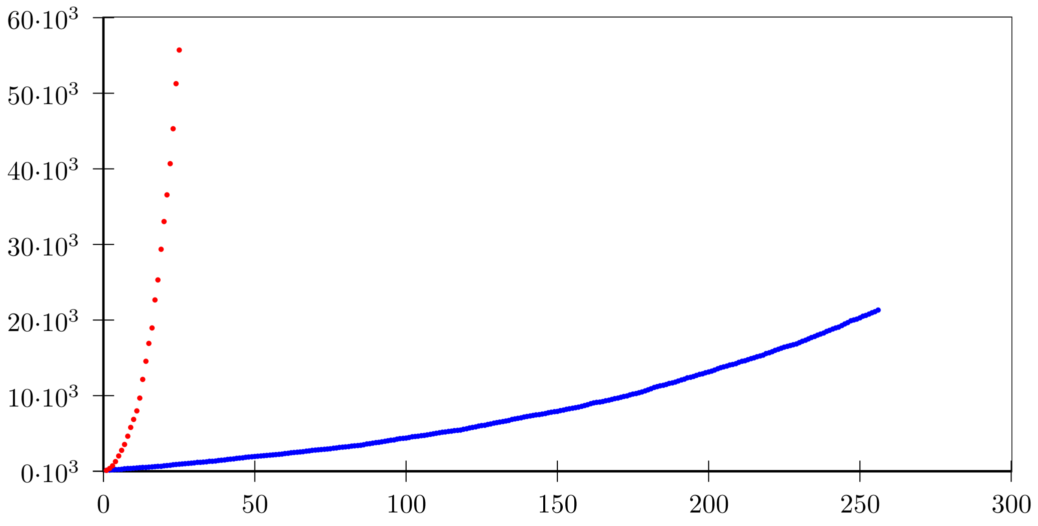

Figure 1: Typical behavior of $\frac{1}{\|z\|^2}$ for $K = \{k\}$ (blue), where $z_k = \min(z)$, and for $K = \{k : z_k \leq 0\}$ (red).

we take for K the set consisting of all indices k for which $z_k \leq 0$, the number of iterations is only 25.

The blue graphs in Figures 1 and 2 show respectively the behavior of $1/\|z\|^2$ and $\max(y)/\sqrt{n}\,\|z\|$ during the course of the algorithm if K is a singleton and the red graphs do the same when we take for K the larger set. Figure 1 makes clear that if K is a singleton the increase in $1/\|z\|^2$ is on average 80, which is much larger than 1, as guaranteed by Lemma 4.1. But if K consists of all k for which $z_k \leq 0$, the increase per iteration is on the average 2200, which is also much larger than guaranteed by Lemma 4.1.

To compare Algorithm 1 and the original BP of Chubanov [1, 2] (with $|K| = 1$ and bound(y) as in (9)) further we also present Table 1. Each line in this tabel gives the average number of iterations, the average time (in seconds) and the average size of the set K, for a class of 100 randomly generated problems with matrices A of size $m \times n$ as given in the first two columns. The elements of A were randomly chosen in the interval $[-100, 100]$, and uniformly distributed. In all cases we used $y = e/n$ as initial vector, where e denotes the all-one vector.

We conclude from Table 1 that in the new approach the average size of the set K is substantially larger than 1. Moreover, as expected, the improvement factor for the average number of iterations is about the same as the average size of K. The improvement factor for the computational time is less predictable. This might be due to the fact that the computation of K and p_K becomes more demanding. The computation of p_K requires $O(|K|\,n)$ time, which is a factor $|K|$ larger than when K is a singleton. We verified that if $m = 625$ then this computation can be responsible for about 80% of the time needed per iteration. This might explain the smaller reduction in computational time. On the other hand, it might also be due to our implementation: the computational time depends on the implementation, which does not hold for the iteration number. A more sophisticated implementation might bring the behavior of the the computational time more in line with that of the iteration number. For further computational evidence in favor of the new

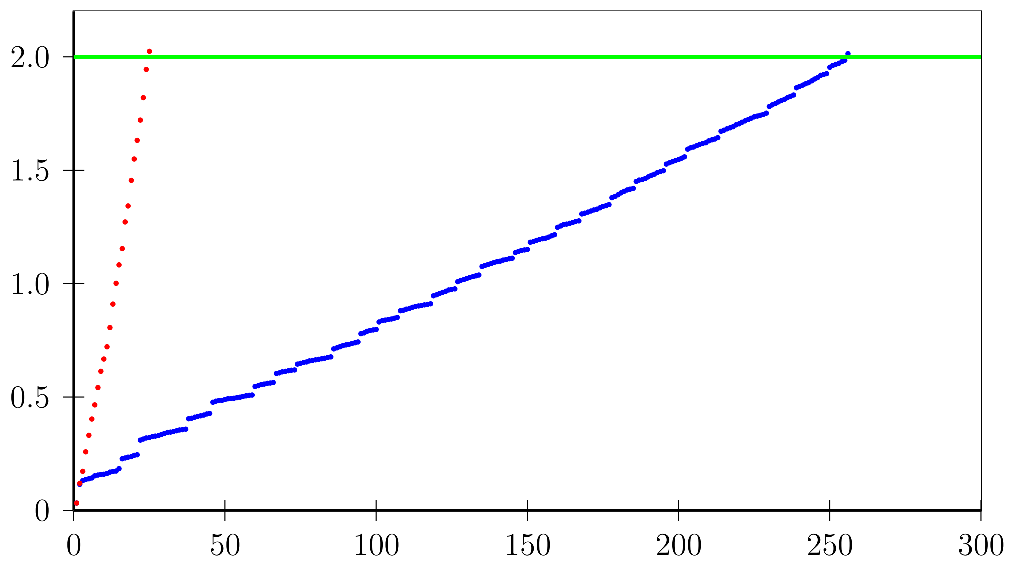

Figure 2: Typical behavior of $\frac{\max(y)}{\sqrt{n}\|z\|}$ for $K = \{k\}$, where $z_k = \min(z)$ (blue), and for $K = \{k : z_k \leq 0\}$ (red).

Table 1: Effect of taking $K = \{k : z_k \leq 0\}$ in comparison with the original Basic Procedure.

		$K = \{k\}$; cut: (9)			$K = \{k : z_k \leq 0\}$; cut: (9)			improvement factor	
m	n	iter.	sec.	$\|K\|$	iter.	sec.	$\|K\|$	iter.	sec.
5	10	15.6	0.0004	1.0	8.6	0.0003	2.2	1.82	1.42
25	50	997.9	0.0270	1.0	113.1	0.0046	8.1	8.83	5.81
125	250	37446.7	1.0624	1.0	1225.7	0.0817	29.3	30.55	13.00
625	1250	2153895.1	146.0467	1.0	16995.3	25.6543	123.7	126.74	5.69

approach we refer to [2].

It should be noted that it is not at all excluded that for the new approach worst-case examples exist where $|K| = 1$ in every iteration. But it can easily be understood that the occurrence of this worst-case behaviour will be a rare event. In fact, during our experiments we encountered this event only for (very) small sizes of A, with $n \leq 10$. On the other hand, it might be worth investigating if it is possible to derive an estimate for the average size of K in the new approach.

We repeated a similar experiment with exactly the same set of randomly generated problems as before, but now using the new cuts, as given by (6). Table 2 shows the results. Also here we see a significant reduction, both for the iteration number and for the computational time.

Next we compare the original BP with the version that incorporates both the new cuts and the larger set K. Table 3 shows the result.

Table 2: Effect of using bound (y) as in (6) in the original Basic Procedure.

		$K=\{k\}$; cut: (9)			$K=\{k\}$; cut: (6)			improvement factor	
m	n	iter.	sec.	$\|K\|$	iter.	sec.	$\|K\|$	iter.	sec.
5	10	15.6	0.0004	1.0	1.2	0.0001	1.0	13.23	7.88
25	50	997.9	0.0270	1.0	20.6	0.0009	1.0	48.32	30.89
125	250	37446.7	1.0624	1.0	1790.9	0.0795	1.0	20.91	13.36
625	1250	2153895.1	146.0467	1.0	65930.0	5.7337	1.0	32.67	25.47

Table 3: Effect of using bound (y) as in (6) and $K=\{k\ :\ z_k\leq 0\}$ in the original Basic Procedure.

		$K=\{k\}$; cut: (9)			$K=\{k:z_k\leq 0\}$; cut: (6)			improvement factor	
m	n	iter.	sec.	$\|K\|$	iter.	sec.	$\|K\|$	iter.	sec.
5	10	15.6	0.0004	1.0	1.2	0.0001	1.3	13.34	7.64
25	50	997.9	0.0270	1.0	8.6	0.0006	6.9	116.44	45.16
125	250	37446.7	1.0624	1.0	137.4	0.0118	23.6	272.58	89.93
625	1250	2153895.1	146.0467	1.0	1574.1	1.6875	88.1	1368.30	86.55

So far we used $K=\{k:z_k\leq 0\}$ for the larger sets K. We repeated the last experiment but now using in K only the indices corresponding to the 20 (or less) smallest nonpositive entries in z, which yields the results in Table 4. As we established before this approach gives the best time bound for the BP, namely $O(n^3)$. It looks as if this is the best approach for larger values of n.

The above results were obtained by using Matlab (version R2014a) on a Windows 7 desktop (Intel(R) Core(TM) i3 CPU, 3.2 GHz), with 8 Gb RAM. For the computation of the projection matrix P_A we used the Matlab commands

```
[m,n] = size(A);
[Y,R] = qr(A',0);
P = eye(n) - Y*Y'.
```

7 Concluding remarks

We have shown that the original BP of Chubanov can be significantly improved by allowing the set K to be larger than a singleton and by using a new type of cuts.

Table 4: Effect of using bound (y) as in (6) and $|K| \leq 20$ in the original Basic Procedure.

		$K = \{k\}$; cut: (9)			$\|K\| \leq 20$; cut: (6)			improvement factor	
m	n	iter.	sec.	$\|K\|$	iter.	sec.	$\|K\|$	iter.	sec.
5	10	15.6	0.0004	1.0	1.2	0.0001	1.3	13.34	3.09
25	50	997.9	0.0270	1.0	8.6	0.0012	6.9	116.44	23.27
125	250	37446.7	1.0624	1.0	142.8	0.0208	15.5	262.20	51.08
625	1250	2153895.1	146.0467	1.0	3684.3	1.2052	19.7	584.62	121.18

It is quite disappointing that in the derivation of the iteration bound we only use that $|K| \geq 1$, whereas in practice $|K|$ is much larger. So it remains as a topic for further research to investigate if the result in Lemma 4.1 can be used to improve the theoretical iteration bound of the BP.

Finally, as Chubanov mentions in [2], his BP resembles a procedure proposed by Von Neumann, which has been described by Dantzig in [3]. This Von Neumann algorithm has been elaborated further in [5] and more recently in [8]. It may be a subject for further research to investigate if the idea developed in the current paper can also be used to speed up Von Neumann's procedure.

Acknowledgement

We thankfully acknowledge valuable comments of an anonymous referee that not only made the paper more readable but also helped to reduce the time for computing the matrix P_A.

References

[1] Sergei Chubanov. A polynomial relaxation-type algorithm for linear programming, 2012. `http://www.optimization-online.org/DB_FILE/2011/02/2915.pdf`.

[2] Sergei Chubanov. A polynomial projection algorithm for linear programming, 2013. `http://www.optimization-online.org/DB_FILE/2013/07/3948.pdf`.

[3] G.B. Dantzig. An ϵ-precise feasible solution to a linear program with a convexity constraint in $1/\epsilon^2$ iterations, independent of problem size. Technical Report SOL 92-5, Systems Optimization Laboratory. Department of Operations Research. Stanford University, Stanford, USA, October 1992.

[4] J. Edmonds. Systems of distinct representatives and linear algebra. *J. Res. Nat. Bur. Standards Sect. B*, 71B:241–245, 1967.

[5] Marina Epelman and Robert M. Freund. Condition number complexity of an elementary algorithm for computing a reliable solution of a conic linear system. *Math. Program.*, 88(3, Ser. A):451–485, 2000.

[6] G.H. Golub and C.F. Van Loan. *Matrix Computations.* Johns Hopkins University Press, Baltimore, second edition, 1989.

[7] L. G. Khachiyan. A polynomial algorithm in linear programming. *Doklady Akademiia Nauk SSSR*, 244:1093–1096, 1979. Translated into English in *Soviet Mathematics Doklady* 20, 191–194.

[8] D. Li and T. Terlaky. The duality between the percepton algorithm and the Von Neumann algorithm. In L. Zukuaga and T. Terlaky, editors, *Modelling and Optimization: Theory and Applications*, pages 113–136, Springer Science+Business, New York, 2013.

[9] J. Renegar. A polynomial-time algorithm, based on Newton's method, for linear programming. *Mathematical Programming*, 40:59–93, 1988.

[10] A. Schrijver. *Theory of Linear and Integer Programming.* John Wiley & Sons, New York, 1986.

[11] E. Stiemke. Über positive Lösungen homogener linearer Gleichungen. *Mathematische Annalen*, 76:340–342, 1915.