TU Delft

Delft University of Technology

Using Nemirovski's Mirror-Prox method as basic procedure in Chubanov's method for solving homogeneous feasibility problems

Wei, Zhang; Roos, Kees

DOI
10.1080/10556788.2021.2023523

Publication date
2022

Document Version
Final published version

Published in
Optimization Methods and Software

Citation (APA)
Wei, Z., & Roos, K. (2022). Using Nemirovski's Mirror-Prox method as basic procedure in Chubanov's method for solving homogeneous feasibility problems. *Optimization Methods and Software*, *37*(4), 1447-1470. https://doi.org/10.1080/10556788.2021.2023523

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

This work is downloaded from Delft University of Technology.
For technical reasons the number of authors shown on this cover page is limited to a maximum of 10.

Optimization Methods and Software

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/goms20

Using Nemirovski's Mirror-Prox method as basic procedure in Chubanov's method for solving homogeneous feasibility problems

Zhang Wei & Kees Roos

To cite this article: Zhang Wei & Kees Roos (2022): Using Nemirovski's Mirror-Prox method as basic procedure in Chubanov's method for solving homogeneous feasibility problems, Optimization Methods and Software, DOI: 10.1080/10556788.2021.2023523

To link to this article: https://doi.org/10.1080/10556788.2021.2023523

© 2022 The Author(s). Published by Informa UK Limited, trading as Taylor & Francis Group.

Published online: 04 Mar 2022.

Submit your article to this journal

Article views: 243

View related articles

View Crossmark data

Full Terms & Conditions of access and use can be found at
https://www.tandfonline.com/action/journalInformation?journalCode=goms20

OPTIMIZATION METHODS & SOFTWARE
https://doi.org/10.1080/10556788.2021.2023523

Taylor & Francis
Taylor & Francis Group

OPEN ACCESS Check for updates

Using Nemirovski's Mirror-Prox method as basic procedure in Chubanov's method for solving homogeneous feasibility problems

Zhang Wei[a] and Kees Roos[b]

[a]Department of Mathematics, South China University of Technology, Guangzhou, People's Republic of China; [b]Department of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, Delft, Netherlands

ABSTRACT

We introduce a new variant of Chubanov's method for solving linear homogeneous systems with positive variables. In the Basic Procedure we use a recently introduced cut in combination with Nemirovski's Mirror-Prox method. We show that the cut requires at most $O(n^3)$ time, just as Chubanov's cut. In an earlier paper it was shown that the new cuts are at least as sharp as those of Chubanov. Our Modified Main Algorithm is in essence the same as Chubanov's Main Algorithm, except that it uses the new Basic Procedure as a subroutine. The new method has $O(n^{3.5}\log_2(1/\delta_A))$ time complexity, where δ_A is a suitably defined condition number. As we show, a simplified version of the new Basic Procedure competes well with the Smooth Perceptron Scheme of Peña and Soheili and, when combined with Rescaling, also with two commercial codes for linear optimization.

ARTICLE HISTORY
Received 25 June 2020
Accepted 20 December 2021

KEYWORDS
Linear optimization; Mirror-Prox method; Chubanov's method; homogeneous

MATHEMATICS SUBJECT CLASSIFICATIONS
90C05; 90C46; 90C47

1. Introduction

We deal with the (primal) problem

$$\begin{array}{ll} \text{find} & x \in \mathbf{R}^n \\ \text{subject to} & Ax = 0, \quad x > 0, \end{array} \tag{1}$$

where A is an integer (or rational) matrix of size $m \times n$ and $\operatorname{rank}(A) = m$. The dual problem is

$$\begin{array}{ll} \text{find} & w \in \mathbf{R}^m \\ \text{subject to} & A^T w \geq 0, \quad A^T w \neq 0. \end{array} \tag{2}$$

According to a variant of Farkas' lemma, due to Stiemke [13], the systems (1) and (2) form an alternative pair in the sense that exactly one of them is feasible [10].

Recently Chubanov [4] proposed a polynomial-time algorithm to deal with this pair of problems. Since the system (1) is homogeneous in x it has a feasible solution if and only

CONTACT Kees Roos c.roos@tudelft.nl

© 2022 The Author(s). Published by Informa UK Limited, trading as Taylor & Francis Group.
This is an Open Access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

if the system

$$Ax = 0, \quad x \in (0,1]^n \tag{3}$$

is feasible. A key ingredient in Chubanov's algorithm is the so-called Basic Procedure (BP). As a result of the BP one gets either

(*i*) a feasible solution of (1), or
(*ii*) a feasible solution for the dual problem (2) of (1), or
(*iii*) a cut for the feasible region of (3).

The cut in (*iii*) has the form $x_k \leq \frac{1}{2}$, for some index k, for all feasible solutions of (3). If this happens, the cut is used to rescale the matrix A. The rescaling happens in Chubanov's Main Algorithm (MA). The MA maintains a vector d such that $x \leq d$ for all x satisfying (3). Initially we take $d = \mathbf{1}$. In case (*iii*) the MA multiplies d_k with $\frac{1}{2}$ and replaces A by AD, where $D = \text{diag}(d)$. Then the MA sends the rescaled matrix to the BP. This is repeated until the BP returns (*i*) or (*ii*). If the BP yields (*ii*) then (1) is infeasible.

Following [9] we use the condition number δ_A, defined by

$$\delta_A := \max\left\{\prod_{i=1}^{n} x_i \ : \ Ax = 0,\ x \in [0,1]^n\right\}.$$

Obviously, $0 \leq \delta_A \leq 1$. Moreover, since $x \leq d$ for all feasible solutions of (3), $\delta_A \leq d_1 d_2 \ldots d_n$. The number of iterations of the MA can now be easily expressed in δ_A, because in case (*iii*) the MA reduces at least one of the entries in d with the factor 2. Hence, after k iterations of the MA we will have $\delta_A \leq 2^{-k}$. As a consequence, $\log_2(1/\delta_A)$ provides an upper bound for the number of iterations of the MA.

The BP in [4] needs at most $4n^3$ iterations per call and $O(n)$ time per iteration, in total $O(n^4)$ time per call. So the overall time complexity becomes $O(n^4 \log_2(1/\delta_A))$.

In [11,12] some improvements of Chubanov's method and its analysis were presented. One improvement was the introduction of a new type of cut. It was shown in [12] that the new cuts are at least as tight as the cuts used by Chubanov. As just mentioned, a Chubanov cut is generated in at most $4n^3$ iterations, whereby an iteration requires $O(n)$ time. The new cuts are also generated in $O(n^3)$ iterations, but this is much more difficult to understand. It was already claimed in a lemma in [12], but unfortunately the proof of this lemma is wrong.

The first result of this paper is in Section 4, where we prove a slightly modified version of [12, Lemma 4.2]. It implies that when the new cuts are used in Chubanov's BP at most n^3 iterations are needed to generate such a cut.

It may be useful to point out that Chubanov's BP is in essence nothing else than an algorithm that in itself is able to solve the problem. In fact it is closely related to the well-known Von Neumann algorithm, having exactly the same convergence rate as that algorithm [5]. The idea underlying Chubanov's method is that such a method can be improved by using information that becomes available during the execution of the algorithm to improve the condition of the underlying matrix A, by rescaling the columns of this matrix. The second result is that Nemirovski's Mirror-Prox method [8] can be used in this way. It generates a cut in only $2n\sqrt{n}$ iterations, with each iteration requiring $O(n^2)$

time. So each call of the new BP needs only $O(n^{3.5})$ time. As a result we obtain that the MA – which is in essence the same as in [4] – finds a solution of either (1) or (2) in $O(n^{3.5}\log_2(1/\delta_A))$ time.

The outline of the paper is as follows. In Section 2 we introduce the notions *inducing primal* and *inducing dual feasibility*, for any vector $y \in \mathbf{R}^n$. We recall in Section 3 how cuts were obtained in [12], as well as some properties of these cuts. The main result in Section 4 is Proposition 4.1; it guarantees that Chubanov's BP generates the new cuts in n^3 iterations.

As a preparation for Section 6, where we present the simple version of Nemirovski's Mirror-Prox method that we use in our BP, we derive in Section 5 the saddle point problem that we want to solve. In Section 7 we present the new BP and its analysis, and in Section 8 the MA. Finally, Section 9 contains some computational results. We compare the method presented in this paper with the smooth perceptron BP presented in [9] and also with the linear optimization codes in two commercial packages: Gurobi [14] and Mosek [1,15]; these packages are freely available for academic use. We conclude with some comments in Section 10.

2. Preliminaries

As usual $\mathbf{R}$ denotes the set of real numbers and $\mathbf{R}_+$ the set of nonnegative real numbers. The all-one vector in $\mathbf{R}^n$ is denoted as $\mathbf{1}$ and the $n \times n$ identity matrix as I_n. The null space of matrix A is denoted as $\mathcal{L}$ and $\mathcal{L}^\perp$ denotes the row space of A. So

$$\mathcal{L} := \left\{x \in \mathbf{R}^n \ : \ Ax = 0\right\}, \quad \mathcal{L}^\perp := \left\{A^T u \ : \ u \in \mathbf{R}^m\right\}. \tag{4}$$

Since rank $(A) = m$, the inverse of AA^T exists. Hence the orthogonal projections P_A and Q_A of $\mathbf{R}^n$ onto $\mathcal{L}$ and $\mathcal{L}^\perp$ are, respectively, given by

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

For any $y \in \mathbf{R}^n$ we use the notation

$$y^{\mathcal{L}} = P_A y, \quad y^{\mathcal{L}^\perp} = Q_A y.$$

So $y^{\mathcal{L}}$ and $y^{\mathcal{L}^\perp}$ are the orthogonal components of y in the spaces $\mathcal{L}$ and $\mathcal{L}^\perp$, respectively:

$$y = y^{\mathcal{L}} + y^{\mathcal{L}^\perp}, \quad y^{\mathcal{L}} \in \mathcal{L}, \quad y^{\mathcal{L}^\perp} \in \mathcal{L}^\perp.$$

Lemma 2.1. *Let $y \in \mathbf{R}^n$. If $y^{\mathcal{L}} > 0$ then $y^{\mathcal{L}}$ solves the primal problem (1) and if $0 \neq y^{\mathcal{L}^\perp} \geq 0$ then $y^{\mathcal{L}^\perp}$ gives rise to a solution of the dual problem (2) in $O(n^3)$ time.*

Proof: Since $y^{\mathcal{L}}$ is the projection of y into the null space of A we have $Ay^{\mathcal{L}} = 0$. Hence the first statement in the lemma is obvious. The second statement follows by noting that $y^{\mathcal{L}^\perp} \in \mathcal{L}^\perp$ implies $y^{\mathcal{L}^\perp} = A^T w$ for some w, thus yielding a solution w of (2). Since A has full row rank, w is uniquely determined by $y^{\mathcal{L}^\perp}$ and can be computed from $y^{\mathcal{L}^\perp}$ in $O(n^3)$ time. ■

Because of Lemma 2.1 it becomes natural to say that a vector *y induces primal feasibility* if $y^{\mathcal{L}} > 0$ and that *y induces dual feasibility* if $0 \neq y^{\mathcal{L}^\perp} \geq 0$. If y does not yield a solution of the primal or the dual problem, it is modified in Chubanov's BP until it induces primal or dual feasibility, or can be used to generate a cut. In the next section we discuss how cuts can be obtained from a vector y, as proposed in [12].

3. Cuts and cutting vectors

While Chubanov used the vector $y^{\mathcal{L}}$ to construct cuts for (3), we showed in [12] that by using the vector $y^{\mathcal{L}^\perp}$ one gets cuts that are at least as tight as the cuts used by Chubanov. Next we recall how this goes.

We introduce the following notations. Let $v = y^{\mathcal{L}^\perp}$. The vector that arises from v by replacing all its negative entries by zero is denoted as v^+. So $v^+ = \max(v, 0)$. Similarly, $v^- = \min(v, 0)$. For each entry v_k of v we define $\sigma_k(v) = 1$ if $v_k = 0$ and otherwise

$$\sigma_k(v) := \mathbf{1}^T \left(\frac{v}{-v_k} \right)^+, \quad v_k \neq 0.$$

In words, if v_k is nonzero then $\sigma_k(v)$ is the sum of the positive entries in the vector $-v/v_k$. Obviously, $\sigma_k(v) \geq 0$ and $\sigma_k(v) = \sigma_k(-v)$. More generally, $\sigma_k(v)$ is homogeneous in v, i.e. $\sigma_k(\beta v) = \sigma_k(v)$ for each $\beta \neq 0$. Moreover, $\sigma_k(v) > 0$ holds if and only if v has entries with a sign opposite to the sign of v_k. Hence, $\sigma_k(v) = 0$ holds if and only if $v_k \neq 0$ and $v \geq 0$ or $v \leq 0$.

We recall the following important lemma [12, Lemma 2.2]; for the sake of completeness we include its simple proof.

Lemma 3.1. *Let x be feasible for (3) and $v \in \mathcal{L}^\perp$. Then every nonzero entry v_k in v gives rise to an upper bound for x_k, according to*

$$x_k \leq \sigma_k(v). \tag{5}$$

Proof: Since $x \in \mathcal{L}$ and $v \in \mathcal{L}^\perp$ we have $v^T x = 0$. Let k be such that $v_k < 0$. Using $v^T x = 0$ and $0 \leq x \leq \mathbf{1}$ we may write

$$-v_k x_k = \sum_{i \neq k} v_i x_i \leq \sum_{i,\, v_i > 0} v_i x_i \leq \sum_{i,\, v_i > 0} v_i = \mathbf{1}^T v^+.$$

After dividing both sides by $-v_k$ we obtain that x_k is bounded from above by the sum of the positive entries in the vector $-v/v_k$. If $v_k > 0$ a similar argument yields the same upper bound for x_k. This proves the lemma. ■

If x is feasible for (3) then we already know that $x_k \leq 1$, for each k. Therefore we say that the cut (5) is void if $\sigma_k(v) \geq 1$. Of course, we are only interested in the nonvoid cases. The following lemma is in the same spirit as Corollary 2.3 in [12], but stronger.

Lemma 3.2. *Let $v_k \neq 0$ and $\sigma_k(v) < 1$ for some k. Then $v_k \mathbf{1}^T v > 0$.*

Proof: First consider the case where $v_k < 0$. In that case $\sigma_k(v) < 1$ holds if and only if $\mathbf{1}^T v^+ < -v_k$. Therefore, we may write

$$\mathbf{1}^T v = \mathbf{1}^T v^+ + \mathbf{1}^T v^- \leq \mathbf{1}^T v^+ + v_k < 0.$$

Similarly, if $v_k > 0$ then $\sigma_k(v) < 1$ holds if and only if $-\mathbf{1}^T v^- < v_k$, whence we get

$$\mathbf{1}^T v = \mathbf{1}^T v^+ + \mathbf{1}^T v^- \geq v_k + \mathbf{1}^T v^- > 0.$$

This proves the lemma. ■

Below it is always assumed that $v = y^{\mathcal{L}^\perp}$ for some y. We define

$$\begin{aligned} \sigma^+(v) &:= \min_k \{\sigma_k(v) \ : \ v_k > 0\} \\ \sigma^-(v) &:= \min_k \{\sigma_k(v) \ : \ v_k < 0\} \\ \sigma(v) &:= \min\left(\sigma^+(v), \sigma^-(v)\right). \end{aligned} \tag{6}$$

Because of Lemma 3.2 $\sigma^+(v) < 1$ can hold only if $\mathbf{1}^T v > 0$ and $\sigma^-(v) < 1$ only if $\mathbf{1}^T v < 0$. So we cannot have both $\sigma^+(v) < 1$ and $\sigma^-(v) < 1$. As a consequence, if $\sigma(v) < 1$ then all v_k's such that $\sigma_k(v) = \sigma(v)$ have the same sign as $\mathbf{1}^T v$. Let us point out that this may be not true if $\sigma(v) \geq 1$. Take, e.g. $v = [1, 1, 1, 1, -2]$. Then $\sigma_k(v) = 2$ for each k.

Lemma 3.3. *Let $v = y^{\mathcal{L}^\perp}$ for some y. If $\sigma(v) = 0$ then problem (1) is infeasible.*

Proof: Let v be as in the lemma and $\sigma(v) = 0$. Then $\sigma_k(v) = 0$ for some k, by (6). By Lemma 3.1 this implies $x_k = 0$ for every x such that $Ax = 0$. This means that (1) is infeasible, proving the lemma. ■

We call y a *cutting vector* if $\sigma(v) < 1$, and a *proper cutting vector* if $\sigma(v) < \frac{1}{2}$.

If y induces primal feasibility then $y^{\mathcal{L}} > 0$. Since $y^{\mathcal{L}} = P_A y^{\mathcal{L}}$ we may restrict our search for a cutting vector to nonnegative vectors y. Due to homogeneity we may further assume $\mathbf{1}^T y = 1$. We conclude this section with a slight modification of a result from [12] that provides a sufficient condition for y being a cutting vector.

Lemma 3.4. *Let $y \geq 0$ satisfy $\mathbf{1}^T y = 1$ and*

$$n^3 \left\| y^{\mathcal{L}} \right\|^2 \leq 1. \tag{7}$$

Then y is a proper cutting vector, i.e. $\sigma(y^{\mathcal{L}^\perp}) < \frac{1}{2}$.

As mentioned in the Introduction, the proof of the original lemma in [12] is wrong. In the next section we provide a proof of Lemma 3.4. As is known, each iteration of Chubanov's BP increases $1/\|y^{\mathcal{L}}\|^2$ with at least 1 [4,11]. Therefore, when Chubanov's BP is equipped with the new cuts, after n^3 iterations (7) certainly holds. It follows that n^3 will be an upper bound for the number of iterations, despite the fact that the new cut is usually tighter than Chubanov's cut and never less tight [11, Section 2.2].

4. Sufficient condition for cutting vectors

In this section $\mathcal{L}$ denotes a fixed linear space in $\mathbf{R}^n$. Moreover, the vector y will be such that $y \geq 0$, $\mathbf{1}^T y = 1$, and z and v are defined by $z = y^{\mathcal{L}}$, $v = y^{\mathcal{L}^\perp}$. Note that if y is not a proper cutting vector, then Lemma 3.4 states that $1 < n^3 \|z\|^2$. More generally, we have the following result.

Proposition 4.1. *If* $\sigma(v) \geq \frac{1}{2}$, *then*

$$1 < n^3 \|z\|^2 \leq n^2.$$

For the proof of this proposition we need a couple of lemmas. We have

$$y \geq 0,\ \mathbf{1}^T y = 1, \quad y = z + v, \quad z^T v = 0, \tag{8}$$

with $\sigma(v) \geq \frac{1}{2}$. The latter implies $n \geq 2$, because $\sigma(v) > 0$ can hold only if v has entries with different signs. In order to derive a lower bound for $\|z\|$ we consider the problem

$$\min_{y,z \in \mathbf{R}^n,\ \beta \in \mathbf{R}} \left\{ \|z\| \ :\ y \geq 0,\ \mathbf{1}^T y = 1,\ y = z + \beta v,\ z^T v = 0 \right\}, \tag{9}$$

with the vector v fixed. We introduced an additional variable β because if $\beta = 1$, as in (8), then problem (9) may be infeasible. This can be understood by noting that if z and v form an orthogonal decomposition of y, as in (8), then $\|v\| \leq \|y\|$. Since $\mathbf{1}^T y = 1$ and $y \geq 0$ we have $\|y\| \leq 1$. So, if $\|v\| > 1$, the system (8) will be infeasible. [1]

The proof below of Proposition 4.1 depends on the fact that we can solve problem (9) analytically. We start with two simple lemmas.

Lemma 4.2. *Let* $\sigma(v) \geq \frac{1}{2}$ *and* (y, z, β) *a feasible solution of* (9). *Then* $z \neq 0$.

Proof: Suppose $z = 0$. Then $y = z + \beta v$ implies $y = \beta v$. Now $0 \neq y \geq 0$. As a consequence $v \neq 0$ and either $v \geq 0$ or $v \leq 0$. So, all nonzero entries of v have the same sign. But then $\sigma(v) = 0$, contradicting $\sigma(v) \geq \frac{1}{2}$. Hence the lemma follows. ■

In order to proceed we need the dual problem of (9), which is given by

$$\max_{\lambda \in \mathbf{R}^n,\ \alpha \in \mathbf{R}} \left\{ \alpha \ :\ \lambda \geq \alpha \mathbf{1},\ \|\lambda\| \leq 1,\ \lambda^T v = 0 \right\}. \tag{10}$$

This problem is strictly feasible (take $\lambda = 0, \alpha = -1$). The dual objective value is bounded above ($\alpha < 1$, because if $\alpha \geq 1$ then $\lambda \geq \mathbf{1}$, whence $\|\lambda\| \geq \sqrt{n}$). These two properties (i.e. strict feasibility and boundedness) imply that (9) is solvable (i.e. has an optimal solution) and that the optimal values of (9) and (10) coincide [2, Thm. 2.4.1]. In the same way it follows that the dual problem is solvable. Hence, there exist optimal solutions (y, z, β) and (λ, α) of (9) and (10), respectively, and $\|z\| = \alpha$. Moreover, by Lemma 4.2, $\|z\| > 0$, whence also $\alpha > 0$. From now on we always assume that the triple (y, z, β) is primal optimal and the pair (λ, α) dual optimal.

Due to the Cauchy-Schwarz inequality and the feasibility conditions we may write

$$\|z\| \geq \|z\| \, \|\lambda\| \geq \lambda^T z = \lambda^T (y - \beta v) = \lambda^T y \geq \alpha \mathbf{1}^T y = \alpha. \tag{11}$$

Since $\|z\| = \alpha$, the three inequalities in (11) hold with equality. Thus we obtain

$$\|z\| = \|z\| \, \|\lambda\|, \quad \|z\| \, \|\lambda\| = \lambda^T z, \quad y^T (\lambda - \alpha \mathbf{1}) = 0. \tag{12}$$

Since $z \neq 0$, by Lemma 4.2, the first equality in (12) implies $\|\lambda\| = 1$ and then the second equation implies $\lambda = z/\|z\|$, whence we obtain

$$z = \alpha \lambda. \tag{13}$$

We derive from $\|\lambda\| = 1$ and the Cauchy-Schwarz inequality that

$$\mathbf{1}^T \lambda \leq \|\mathbf{1}\| \, \|\lambda\| = \|\mathbf{1}\| = \sqrt{n}. \tag{14}$$

Lemma 4.3. *We always have* $n \|z\|^2 \leq 1$. *Equality holds if and only if* $\mathbf{1}^T v = 0$.

Proof: Using $\mathbf{1}^T \mathbf{1} = n$, $\lambda \geq \alpha \mathbf{1}$ and (14), we may write

$$\alpha n = \mathbf{1}^T (\alpha \mathbf{1}) \leq \mathbf{1}^T \lambda \leq \sqrt{n}. \tag{15}$$

Therefore, $\alpha \sqrt{n} \leq 1$. Since $\|z\| = \alpha$, the first statement in the lemma follows.

Next we deal with the second statement. It can be restated as $n\alpha^2 = 1$ holds if and only if $\mathbf{1}^T v = 0$. First assume $n\alpha^2 = 1$. Then (15) implies $\mathbf{1}^T \lambda = \sqrt{n}$, whence (14) implies $\mathbf{1}^T \lambda = \|\mathbf{1}\| \, \|\lambda\|$. Since $\|\lambda\| = 1$ this gives $\lambda = \mathbf{1}/\sqrt{n}$. Since λ is dual feasible, we have $\lambda^T v = 0$, and hence also $\mathbf{1}^T v = 0$. On the other hand, if $\mathbf{1}^T v = 0$ then $y = z + \beta v$ and $\mathbf{1}^T y = 1$ imply $\mathbf{1}^T z = 1$. This implies $\|\mathbf{1}\| \, \|z\| \geq 1$, whence $n \|z\|^2 \geq 1$. Due to the first statement in the lemma and $\alpha = \|z\|$ we obtain $n\alpha^2 = 1$, thereby completing the proof of the lemma. ■

The inequality $n \|z\|^2 \leq 1$ in Lemma 4.3 is equivalent with the inequality at the right in Proposition 4.1. The proof of the left inequality in Proposition 4.1 requires in general much more work, but not if $\mathbf{1}^T v = 0$. Then Lemma 4.3 yields $n \|z\|^2 = 1$. Since $n > 1$ this implies $n^3 \|z\|^2 = n^2 \cdot n \|z\|^2 = n^2 > 1$. So, for the rest of proof we may assume $\mathbf{1}^T v \neq 0$.

In the sequel I will denote the subset of the index set $\{1, 2, \ldots, n\}$ defined by

$$I = \{i \, : \, y_i > 0\}, \tag{16}$$

and J its complement. The restriction of $v \in \mathbf{R}^n$ to the coordinates in I is denoted as v_I. Using these notations we have the following lemma.

Lemma 4.4. *Let* v *be such that* $\sigma(v) \geq \frac{1}{2}$ *and* $\mathbf{1}^T v \neq 0$. *Then*

$$\beta = \frac{\alpha^2 \mathbf{1}_I^T v_I}{\|v_J\|^2}, \quad \alpha^2 = \frac{\|v_J\|^2}{|I| \, \|v_J\|^2 + \left(\mathbf{1}_I^T v_I\right)^2}. \tag{17}$$

Proof: We already derived from the first two equalities in (12) that $\|\lambda\| = 1$ and, in (13), $z = \alpha\lambda$. Since $y \geq 0$ and $\lambda - \alpha\mathbf{1} \geq 0$, the third equality in (12) implies $y_i(\lambda_i - \alpha) = 0$, for each i. Therefore, due to the definition of I, $\lambda_i = \alpha$ for each $i \in I$. So we have

$$\lambda_I = \alpha\mathbf{1}_I, \quad \lambda_J \geq \alpha\mathbf{1}_J,$$

where the inequality is due to $\lambda \geq \alpha\mathbf{1}$. From $y_J = 0$ we deduce $z_J = -\beta v_J$. Also using $z = \alpha\lambda$ we get $z_I = \alpha\lambda_I = \alpha^2\mathbf{1}_I$. Partitioning the vectors v, y, z and λ according to the partition (I, J), we get the following expressions:

$$v = \begin{bmatrix} v_I \\ v_J \end{bmatrix}, \quad y = \begin{bmatrix} \alpha^2\mathbf{1}_I + \beta v_I \\ 0 \end{bmatrix}, \quad z = \begin{bmatrix} \alpha^2\mathbf{1}_I \\ -\beta v_J \end{bmatrix}, \quad \lambda = \begin{bmatrix} \alpha\mathbf{1}_I \\ -\frac{\beta}{\alpha}v_J. \end{bmatrix} \tag{18}$$

Now the primal feasibility conditions $\mathbf{1}^T y = 1$ and $z^T v = 0$ yield two linear relations between α^2 and β, as follows:

$$1 = \mathbf{1}^T y = \mathbf{1}_I^T y_I = \mathbf{1}_I^T\left(\alpha^2\mathbf{1}_I + \beta v_I\right) = \alpha^2 |I| + \beta\mathbf{1}_I^T v_I,$$
$$0 = z^T v = z_I^T v_I + z_J^T v_J = \alpha^2\mathbf{1}_I^T v_I + (-\beta v_J)^T v_J = \alpha^2\mathbf{1}_I^T v_I - \beta \left\|v_J\right\|^2.$$

The determinant of the matrix of coefficients of this linear system of equations equals $(\mathbf{1}_I^T v_I)^2 + |I| \left\|v_J\right\|^2$. This expression is positive, because otherwise we would have $\mathbf{1}_I^T v_I = 0$ and $v_J = 0$, which would imply $\mathbf{1}^T v = 0$, a contradiction with the hypothesis in the lemma. Hence the solution of the above system is unique. It is given by (17). This completes the proof. ■

Obviously, we can compute α and β from (17) if we know the 'optimal' partition (I, J) of the index set, since α is positive. After this the optimal solution immediately follows from (18). In order to proceed we need to know more about this partition. For that purpose the next lemma is important.

Lemma 4.5. *With v as in Lemma 4.4, let (y, z, β) be optimal for (9). Then $\beta\mathbf{1}^T v > 0$.*

Proof: Taking the inner product with $\mathbf{1}$ at both sides of $y = z + \beta v$ we may write

$$1 = \mathbf{1}^T y = \mathbf{1}^T z + \beta\mathbf{1}^T v = \alpha\mathbf{1}^T\lambda + \beta\mathbf{1}^T v \leq \alpha\sqrt{n} + \beta\mathbf{1}^T v < 1 + \beta\mathbf{1}^T v,$$

where we used $z = \alpha\lambda$, by (13), $\mathbf{1}^T\lambda \leq \sqrt{n}$, from (14), and $\alpha\sqrt{n} < 1$, by Lemma 4.3. This suffices for the proof of the lemma. ■

If we replace v by $-v$, and β by $-\beta$, then (y, z, β) and (λ, α) stay primal and dual optimal, respectively. We may therefore assume that $\mathbf{1}^T v$ is nonnegative, without loss of generality. Since we only need to deal with the case where $\mathbf{1}^T v \neq 0$, we assume in the sequel that $\mathbf{1}^T v > 0$. Because of Lemma 4.5 we then have

$$\mathbf{1}^T v > 0, \quad \beta > 0. \tag{19}$$

Yet we take into account the feasibility conditions $y_I > 0$ and $\lambda_J \geq \alpha\mathbf{1}_J$. By (18) these conditions require $\alpha^2\mathbf{1}_I + \beta v_I > 0$ and $-\frac{\beta}{\alpha}v_J \geq \alpha\mathbf{1}_J$. Since $\alpha > 0$, these inequalities can be reformulated as follows:

$$\alpha^2 + \beta v_i > 0, \quad i \in I,$$

$$\alpha^2 + \beta v_j \leq 0, \quad j \in J.$$

Since $\beta > 0$, this makes clear that the entries of v in v_I are strictly larger than those in v_J. Moreover, the entries in v_J are negative, and they are separated from the entries in v_I by the (negative!) number $-\alpha^2/\beta$. Due to Lemma 4.4 this leads to the inequalities

$$v_i > -\frac{\|v_J\|^2}{\mathbf{1}_I^T v_I} \geq v_j, \quad i \in I, \quad j \in J. \tag{20}$$

At this stage it becomes natural to order the entries of v in nonincreasing order. Since v has entries of different signs, there must exist a (unique) index p such that, after the ordering of v,

$$v_1 \geq v_2 \geq \ldots \geq v_p \geq 0 > v_{p+1} \geq \ldots \geq v_n.$$

Moreover, since $v_J < 0$, we have for some $q \geq p$:

$$I = \{1, \ldots, q\}, \quad J = \{q+1, \ldots, n\}.$$

Note that $q \geq p$ implies $v_{q+1} < 0$. Now (20) holds if and only if

$$v_q > -\frac{\|v_J\|^2}{\mathbf{1}_I^T v_I} \geq v_{q+1}. \tag{21}$$

Next we show not only that (21) determines q uniquely, and hence also I and J, but also that q can be found in $O(n)$ time. For that purpose we define, for each k ($1 \leq k \leq n$), index sets $I_k := \{1, \ldots, k\}$ and $J_k := \{k+1, \ldots, n\}$, and numbers ω_k, according to

$$\omega_k := \frac{\|v_{J_k}\|^2}{\mathbf{1}_{I_k}^T v_{I_k}} = \frac{\sum_{i=k+1}^n v_i^2}{\sum_{i=1}^k v_i}.$$

With this definition, (21) can be restated as

$$v_q > -\omega_q \geq v_{q+1}. \tag{22}$$

Observe that the vector ω is nonnegative and $\omega_n = 0$. Moreover, the expressions $\sum_{i=1}^k v_i$ and $\sum_{i=k+1}^n v_i^2$ can be computed from v in $O(k)$ time. As a consequence, the vector ω can be computed in $O(n)$ time. The following lemma determines the index q uniquely.

Lemma 4.6. *q is the first index such that*

$$\omega_q = \max_k \{\omega_k \ : \ p \leq k \leq n\}. \tag{23}$$

Proof: For $k < n$ one may easily verify that

$$\omega_{k+1} - \omega_k = \frac{-v_{k+1}(\omega_k + v_{k+1})}{\sum_{i=1}^{k+1} v_i} = \frac{-v_{k+1}(\omega_{k+1} + v_{k+1})}{\sum_{i=1}^k v_i}.$$

Let $k \geq p$. Then $v_{k+1} < 0$. Since $\sum_{i=1}^k v_i > 0$ for each k, we find that if $k < n$ then

$$\omega_{k+1} > \omega_k \Leftrightarrow \omega_k + v_{k+1} > 0, \tag{24}$$

$$\Leftrightarrow \omega_{k+1} + v_{k+1} > 0. \tag{25}$$

Because of (22) we have $\omega_q + v_q > 0$ and $\omega_q + v_{q+1} \leq 0$. Hence (25) implies $\omega_q > \omega_{q-1}$ and (24) implies $\omega_{q+1} \leq \omega_q$. Thus it becomes clear that ω is increasing at $k = q - 1$ and nonincreasing at $k = q$.

Now suppose that ω_k is nonincreasing at some index $k \geq p$. Then $\omega_{k+1} \leq \omega_k$. Due to (25) we then have $\omega_{k+1} + v_{k+1} \leq 0$. Since $v_{k+2} \leq v_{k+1}$ it follows that $\omega_{k+1} + v_{k+2} \leq 0$, which means that $\omega_{k+2} \leq \omega_{k+1}$, by (24). So, if ω_k is nonincreasing at $k \geq p$, then ω_k remains nonincreasing if k increases. As we proved that ω_k is nonincreasing at $k = q$, it follows that ω_k is nonincreasing at all $k \geq q$. If ω_k were nonincreasing at some $k < q$, it would yield the contradiction that ω_k is nonincreasing at $k = q - 1$. Hence ω_k must be increasing (strictly!) at all k such that $p \leq k < q$. Since ω_k is nonincreasing at all $k \geq q$, the lemma follows. ■

Example 4.7. *By way of example we demonstrate in Table 1 the computation of p and q for the case where*

$$v = [5;\ 4;\ 3;\ 2;\ 1;\ -1;\ -2;\ -2;\ -3;\ -4].$$

In this case $p = 5$. For $k \geq p$ the largest value of ω_k occurs at $k = 8$. So $q = 8$. In agreement with (24) and (25) the table demonstrates that $\omega_{k+1} + v_{k+1}$ and $\omega_k + v_{k+1}$ have the same sign, for each k. Observe that q is the first index for which these expressions are negative.

Table 1. Computation of q.

k	v_k	ω_k	$\omega_k + v_{k+1}$	$\omega_{k+1} + v_{k+1}$
1	5.0000	12.8000	16.8000	9.3333
2	4.0000	5.3333	8.3333	6.2500
3	3.0000	3.2500	5.2500	4.5000
4	2.0000	2.5000	3.5000	3.2667
p = 5	1.0000	2.2667	1.2667	1.3571
6	−1.0000	2.3571	0.3571	0.4167
7	−2.0000	2.4167	0.4167	0.5000
q = 8	−2.0000	2.5000	−0.5000	−0.7143
9	−3.0000	2.2857	−1.7143	−4.0000
10	−4.0000	0.0000	–	–

Lemma 4.8. *With v, α and I as defined above, one has* [2]

$$\alpha^2 \geq \frac{\left\|v^-\right\|^2}{q\left\|v^-\right\|^2 + \left(\mathbf{1}^T v^+\right)^2}.$$

Proof: As before, $I = \{1, \ldots, q\}$. Because of Lemma 4.4, the inequality in the lemma holds if and only if

$$\frac{\left\|v_J\right\|^2}{|I|\left\|v_J\right\|^2 + \left(\mathbf{1}_I^T v_I\right)^2} \geq \frac{\left\|v^-\right\|^2}{q\left\|v^-\right\|^2 + \left(\mathbf{1}^T v^+\right)^2}.$$

Since $|I| = q$ this is equivalent to

$$\frac{\|v_J\|^2}{\left(\mathbf{1}_I^T v_I\right)^2} \geq \frac{\|v^-\|^2}{\left(\mathbf{1}^T v^+\right)^2},$$

which can be written as

$$\left(\mathbf{1}^T v^+\right) \omega_q \geq \left(\mathbf{1}_I^T v_I\right) \omega_p.$$

By Lemma 4.6 we have $\omega_q \geq \omega_p > 0$. Since $\mathbf{1}^T v^+ \geq \mathbf{1}_I^T v_I > 0$, the lemma follows. ■

Now we are ready to show that $n^3\alpha^2 > 1$, which will complete the proof of Proposition 4.1. According to Lemma 4.8 we have

$$\frac{1}{\alpha^2} \leq q + \frac{\left(\mathbf{1}^T v^+\right)^2}{\|v^-\|^2}.$$

The largest element in v is v_1 and $v_1 > 0$. So $\mathbf{1}^T v^+ \leq p v_1$. By the Cauchy-Schwarz inequality, $(n-p)\|v^-\|^2 \geq \left(\mathbf{1}^T v^-\right)^2$. Definition (6) implies $\sigma^+(v) = \sigma_1(v) = -\mathbf{1}^T v^- / v_1$. We therefore may write

$$\frac{\left(\mathbf{1}^T v^+\right)^2}{\|v^-\|^2} \leq \frac{(p v_1)^2}{\|v^-\|^2} \leq (n-p)\frac{(p v_1)^2}{\left(\mathbf{1}^T v^-\right)^2} = \frac{(n-p)p^2}{\sigma^+(v)^2} \leq \frac{(n-p)p^2}{\sigma(v)^2} \leq \frac{4n^3}{27\sigma(v)^2},$$

where we also used $\sigma^+(v) \geq \sigma(v)$ and that $(n-p)p^2$ is maximal if $p = 2n/3$. Since $\sigma(v) \geq \frac{1}{2}$ and $q \leq n$ we thus obtain

$$\frac{1}{\alpha^2} \leq q + \frac{16n^3}{27} \leq n + \frac{16n^3}{27}.$$

The last expression is smaller than n^3 if and only if $11n^2 > 27$ and this certainly holds, because $n \geq 2$. Hence the proof of Proposition 4.1 is complete.

5. A bilinear saddle point problem

In this section we derive a simple saddle point problem that arises when searching for a cutting vector. In the next section we describe Nemirovski's mirror-prox method for solving that saddle point problem.

According to Proposition 4.1, y is a cutting vector if it satisfies $y \geq 0$, $\mathbf{1}^T y = 1$ and $\|y^{\mathcal{L}}\| \leq 1/n\sqrt{n}$, where $\mathcal{L}$ denotes the null space of A. Since $y^{\mathcal{L}} = P_A y$, we therefore consider the (primal) problem

$$\min\left\{\|P_A y\| \;:\; y \geq 0,\ \mathbf{1}^T y = 1\right\}. \tag{26}$$

This is a second-order cone problem again, just as the problem considered in the previous section. Its dual problem is given by

$$\max\{\alpha \;:\; P_A u \geq \alpha \mathbf{1},\ \|u\| \leq 1\}. \tag{27}$$

Now let y^* and u^* denote optimal solutions of (26) and (27), respectively. Then the optimal value of α is given by $\alpha^* = \min(P_A u^*)$.

Lemma 5.1. *If $\alpha^* > 0$, then $P_A u^*$ solves (1). Otherwise, if $\alpha^* = 0$, then y^* solves (2).*

Proof: Note that if (u, α) is feasible for (27) and α positive then $P_A u$ is a positive vector in the null space of A. So, in that case $x = P_A u$ is a solution of (1). On the other hand, if no such pair (u, α) with $\alpha > 0$ exists, then the optimal value of (27) equals 0. Since the problems (26) and (27) have bounded feasible regions, they have optimal solutions and their optimal values are equal. So, problem (26) has optimal value 0 as well and this value is attained. Hence there exists a nonzero nonnegative vector y such that $P_A y = 0$. The latter means that $0 \neq y \in \mathcal{L}^\perp$ and $y \geq 0$. By Lemma 2.1 this implies that (2) has a solution. Since (2) has a solution if and only if (1) has no solution, the lemma follows. ■

For a given u the best value of α in (27) equals the smallest entry in $P_A u$. We therefore assume below that α always satisfies $\alpha = \min(P_A u)$. Then the feasible regions of (26) and (27) are, respectively, the unit simplex Δ and the unit sphere $\mathcal{B}$, as defined by

$$\Delta = \left\{ y \in \mathbf{R}^n \ : \ \mathbf{1}^T y = 1, \, y \geq 0 \right\}, \quad \mathcal{B} = \left\{ u \in \mathbf{R}^n \ : \ \|u\| \leq 1 \right\}.$$

If $y \in \Delta$ and $u \in \mathcal{B}$ then we may write

$$\left\| P_A y \right\| \geq \left\| P_A y \right\| \|u\| \geq y^T P_A u \geq y^T (\alpha \mathbf{1}) = \alpha \mathbf{1}^T y = \alpha = \min(P_A u). \tag{28}$$

Putting $y = y^*$ in (28) it follows that ${y^*}^T P_A u \leq \left\| P_A y^* \right\|$ for each $u \in \mathcal{B}$. Similarly, putting $u = u^*$ in (28) we get $y^T P_A u^* \geq \min(P_A u^*)$ for each $y \in \Delta$. At optimality the primal and dual objective values are equal. So we have $\min(P_A u^*) = \left\| P_A y^* \right\| = {y^*}^T P_A u^*$. Thus we obtain

$${y^*}^T P_A u \leq {y^*}^T P_A u^* \leq y^T P_A u^*, \quad \forall \, y \in \Delta, \quad \forall \, u \in \mathcal{B}. \tag{29}$$

This reveals that $\left(y^*, u^*\right)$ is a saddle point for the bilinear function $y^T P_A u$. Thus we have reduced the solution of (26) and (27) to the computation of the saddle point of $y^T P_A u$.

6. Nemirovski's Mirror-Prox method

6.1. Definition of the method

To simplify notation, from now on we denote P_A simply as P. Following [8], we define the vector field $F(z)$, where $z = (y, u) \in \Delta \times \mathcal{B}$, by

$$F(z) = \begin{bmatrix} \frac{\partial}{\partial y} y^T P_A u \\ -\frac{\partial}{\partial u} y^T P_A u \end{bmatrix} = \begin{bmatrix} Pu \\ -Py \end{bmatrix}.$$

The Mirror-Prox method can now be stated as follows. It is initialized with

$$z_1 = (y_1, u_1) \in \Delta \times \mathcal{B},$$

and uses the following update in each iteration:

$$\hat{z}_k = \operatorname{argmin}_{z \in \Delta \times \mathcal{B}} \left\{ \gamma F(z_k)^T z + \tfrac{1}{2} \|z - z_k\|^2 \right\} \tag{30}$$

$$z_{k+1} = \text{argmin}_{z\in\Delta\times\mathcal{B}} \left\{ \gamma F(\hat{z}_k)^T z + \tfrac{1}{2} \|z - z_k\|^2 \right\}, \tag{31}$$

where γ is a fixed positive 'step size', with $\gamma \in (0, 1]$.

Below we present the analysis of this method. Not surprisingly, due to the linearity in y and u the analysis goes easier than in [8], but the resulting iteration bound is the same.

6.2. Analysis of the method

Lemma 6.1. *For any $z, \bar{z} \in \Delta \times \mathcal{B}$ we have*

(i) $F(z)^T \bar{z} = -F(\bar{z})^T z$;
(ii) $(F(z) - F(\bar{z}))^T (z - \bar{z}) = 0$;
(iii) $\|F(z) - F(\bar{z})\|_2 \leq \|z - \bar{z}\|_2$;

Proof: With $z = (y; u)$ and $\bar{z} = (\bar{y}; \bar{u})$ we have

$$F(z)^T \bar{z} = \begin{bmatrix} Pu \\ -Py \end{bmatrix}^T \begin{bmatrix} \bar{y} \\ \bar{u} \end{bmatrix} = \bar{y}^T Pu - \bar{u}^T Py = \begin{bmatrix} -P\bar{u} \\ P\bar{y} \end{bmatrix}^T \begin{bmatrix} y \\ u \end{bmatrix} = -F(\bar{z})^T z,$$

proving (i). Taking $\bar{z} = z$ in (i) we get $F(z)^T z = 0$ for each z. As a consequence we have

$$(F(z) - F(\bar{z}))^T (z - \bar{z}) = F(z)^T z - F(z)^T \bar{z} - F(\bar{z})^T z + F(\bar{z})^T \bar{z} = 0,$$

proving (ii). Since P is an orthogonal projection matrix, we may write

$$\|F(z) - F(\bar{z})\|_2 = \left\| \begin{bmatrix} P(u - \bar{u}) \\ -P(y - \bar{y}) \end{bmatrix} \right\|_2 \leq \left\| \begin{bmatrix} u - \bar{u} \\ y - \bar{y} \end{bmatrix} \right\|_2 = \|z - \bar{z}\|_2 , \tag{32}$$

proving (iii). ■

To measure the distance between z and $\bar{z}$ we introduce the notation

$$\delta(z, \bar{z}) := \tfrac{1}{2} \|z - \bar{z}\|^2 . \tag{33}$$

Lemma 6.2. *Let $u, v, w \in \Delta \times \mathcal{B}$. Then*

$$(F(u) - F(v))^T (u - w) \leq \delta(u, v) + \delta(u, w).$$

Proof: By the Cauchy-Schwarz inequality and Lemma 6.1 we get

$$(F(u) - F(v))^T (u - w) \leq \|F(u) - F(v)\| \, \|u - w\| \leq \|u - v\| \, \|u - w\| .$$

The last expression does not exceed $\frac{1}{2} \|u - v\|^2 + \frac{1}{2} \|u - w\|^2$. Hence, due to definition (33) the lemma follows. ■

For any three vectors u, v and w of the same dimension we have the trivial identity $\|w - v\|^2 = \|u - v - (u - w)\|^2$. Evaluation of the right-hand side yields the so-called three-point relation of Bregman:

$$(u - v)^T (u - w) = \delta(u, v) + \delta(u, w) - \delta(v, w). \tag{34}$$

Lemma 6.3. *One has the following three inequalities:*

$$\begin{array}{lll} (i) & \gamma\left(F(\hat{z}_k) - F(z_k)\right)^T(\hat{z}_k - z_{k+1}) & \leq \delta(\hat{z}_k, z_{k+1}) + \delta(\hat{z}_k, z_k), \\ (ii) & \gamma F(z_k)^T(\hat{z}_k - z_{k+1}) & \leq -\delta(\hat{z}_k, z_{k+1}) - \delta(\hat{z}_k, z_k) + \delta(z_{k+1}, z_k), \\ (iii) & \gamma F(\hat{z}_k)^T(z_{k+1} - z) & \leq \delta(z, z_k) - \delta(z, z_{k+1}) - \delta(z_{k+1}, z_k). \end{array}$$

Proof: Since $\gamma \leq 1$, (i) follows immediately from Lemma 6.2. The gradient with respect to z of $\gamma F(z_k)^T z + \delta(z, z_k)$ is equal to $\gamma F(z_k) + z - z_k$. Hence optimality of $\hat{z}_k$ in (30) implies

$$\left(\gamma F(z_k) + \hat{z}_k - z_k\right)^T(z - \hat{z}_k) \geq 0, \quad \forall\, z \in \Delta \times \mathcal{B}.$$

Using (34) this gives

$$\gamma F(z_k)^T(\hat{z}_k - z) \leq \left(\hat{z}_k - z_k\right)^T(z - \hat{z}_k) = \delta(z, z_k) - \delta(z, \hat{z}_k) - \delta(\hat{z}_k, z_k).$$

Letting $z = z_{k+1}$, we get (ii). Finally, the gradient with respect to z of $\gamma F(\hat{z}_k)^T z + \delta(z, z_k)$ is equal to $\gamma F(\hat{z}_k) + z - z_k$. Hence optimality of z_{k+1} in (31) implies

$$\left(\gamma F(\hat{z}_k) + z_{k+1} - z_k\right)^T(z - z_{k+1}) \geq 0, \quad \forall\, z \in \Delta \times \mathcal{B},$$

which is equivalent to

$$\gamma F(\hat{z}_k)^T(z_{k+1} - z) \leq \left(z_{k+1} - z_k\right)^T(z - z_{k+1}).$$

By applying Bregman's formula, we get (iii). ■

By adding the three inequalities in Lemma 6.3 we get the following lemma, without further proof (cf. [6, Theorem 18.2]).

Lemma 6.4.

$$\gamma F(\hat{z}_k)^T(\hat{z}_k - z) \leq \delta(z, z_k) - \delta(z, z_{k+1}), \quad \forall\, z \in \Delta \times \mathcal{B}.$$

For $K = 1, 2, \ldots$ we define

$$\tilde{z}_K := (\tilde{y}_K; \tilde{u}_K) = \frac{1}{K}\sum_{k=1}^{K} \hat{z}_k. \tag{35}$$

So $\tilde{z}_K$ is simply the average of all iterates $\hat{z}_k$ with $1 \leq k \leq K$. We denote the *duality gap* at any pair (y, u) with $y \in \Delta$ and $u \in \mathcal{B}$ as

$$\text{gap}(y, u) := \left\|P_A y\right\| - \min(P_A u). \tag{36}$$

The next theorem is the main result in this section. It says that the duality gap at $\tilde{z}_K$ is inversely proportional to the iteration number K.

Theorem 6.5. *By taking $y_1 = \mathbf{1}/n$ and $u_1 = 0$ we obtain*

$$\text{gap}(\tilde{y}_K, \tilde{u}_K) = \left\|P\tilde{y}_K\right\| - \min(P\tilde{u}_K) \leq \frac{1}{K\gamma}, \quad K \geq 1.$$

Proof: Recall that $F(z)^T z = 0$ for every $z \in \Delta \times \mathcal{B}$. Using this we may write

$$F(\hat{z}_k)^T(\hat{z}_k - z) = -F(\hat{z}_k)^T z = -\begin{bmatrix} P\hat{u}_k \\ -P\hat{y}_k \end{bmatrix}^T \begin{bmatrix} y \\ u \end{bmatrix} = -y^T P\hat{u}_k + \hat{y}_k^T Pu,$$
$$\forall\, z = (y,u) \in \Delta \times \mathcal{B}.$$

Hence we obtain from Lemma 6.4 that for each $k = 1, 2, \ldots, K$,

$$\gamma\left(-y^T P\hat{u}_k + \hat{y}_k^T Pu\right) \le \delta(z, z_k) - \delta(z, z_{k+1}), \quad \forall\, z = (y,u) \in \Delta \times \mathcal{B}.$$

By taking the sum of these inequalities for $k = 1, \ldots, K$, and then dividing by $K\gamma$ we get

$$\frac{1}{K\gamma}\sum_{k=1}^{K} \gamma\left(\hat{y}_k^T Pu - y^T P\hat{u}_k\right) \le \frac{1}{K\gamma}\sum_{k=1}^{K}\left(\delta(z, z_k) - \delta(z, z_{k+1})\right) \le \frac{\delta(z, z_1)}{K\gamma}.$$

Since $y_1 = \mathbf{1}/n$ and $u_1 = 0$ we have

$$\max_{z\in\Delta\times\mathcal{B}} \delta(z, z_1) = \tfrac{1}{2}\left(\max_{y\in\Delta}\left\|y - y_1\right\|^2 + \max_{u\in\mathcal{B}}\|u - u_1\|^2\right) \le \tfrac{1}{2}\left(\frac{n-1}{n} + 1\right) < 1.$$

Thus we obtain

$$\frac{1}{K}\sum_{k=1}^{K}\left(\hat{y}_k^T Pu - y^T P\hat{u}_k\right) \le \frac{1}{K\gamma}, \quad \forall\, z = (y,u) \in \Delta \times \mathcal{B}.$$

Using that $y^T Pu$ is linear in both y and u we may write the expression at the left as follows:

$$\frac{1}{K}\sum_{k=1}^{K}\left(\hat{y}_k^T Pu - y^T P\hat{u}_k\right) = \left(\frac{1}{K}\sum_{k=1}^{K}\hat{y}_k\right)^T Pu - y^T P\left(\frac{1}{K}\sum_{k=1}^{K}\hat{u}_k\right) = \tilde{y}_K^T Pu - y^T P\tilde{u}_K.$$

We take $u = P\tilde{y}_K / \left\|P\tilde{y}_K\right\|$ and $y = e_i$, where i is the entry for which $P\tilde{u}_K$ is minimal. Then the expression at the left takes the value $\mathrm{gap}(\tilde{y}_K, \tilde{u}_K)$. Hence the inequality in the lemma follows. ■

7. The Mirror-Prox basic procedure

We now are ready to describe the new BP, called Mirror-Prox BP (abbr. MPBP). Each iteration of the MPBP requires the computation of $\hat{z}_k$ and z_{k+1}, as given by (30) and (31). We claim that this can be done in $O(n^2)$ time. To understand this let $z = (y;u)$, $z_k = (y_k;u_k)$ and $\hat{z}_k = (\hat{y}_k;\hat{u}_k)$. Since $F(z_k) = (Pu_k; -Py_k)$ we may rewrite (30) as follows

$$\begin{aligned}\hat{z}_k &= \mathrm{argmin}_{z\in\Delta\times\mathcal{B}}\left\{\gamma F(z_k)^T z + \tfrac{1}{2}\|z - z_k\|^2\right\}\\ &= \mathrm{argmin}_{y\in\Delta,\, u\in\mathcal{B}}\left\{\gamma\left(y^T Pu_k - u^T Py_k\right) + \tfrac{1}{2}\left\|y - y_k\right\|^2 + \tfrac{1}{2}\|u - u_k\|^2\right\}.\end{aligned}$$

Obviously the computation of $\hat{z}_k$ can be separated into the computation of its components $\hat{y}_k$ and $\hat{u}_k$, as follows:

$$\hat{y}_k = \text{argmin}_{y\in\Delta} \left\{ \gamma y^T P u_k + \tfrac{1}{2} \left\| y - y_k \right\|^2 \right\} \tag{37}$$

$$\hat{u}_k = \text{argmin}_{u\in\mathcal{B}} \left\{ -\gamma u^T P y_k + \tfrac{1}{2} \left\| u - u_k \right\|^2 \right\}. \tag{38}$$

In a similar way problem (31) can be split into the simpler problems

$$y_{k+1} = \text{argmin}_{y\in\Delta} \left\{ \gamma y^T P \hat{u}_k + \tfrac{1}{2} \left\| y - y_k \right\|^2 \right\} \tag{39}$$

$$u_{k+1} = \text{argmin}_{u\in\mathcal{B}} \left\{ -\gamma u^T P \hat{y}_k + \tfrac{1}{2} \left\| u - u_k \right\|^2 \right\}. \tag{40}$$

So, given a pair (y_k, u_k), the solution of these four problems yields the pair (y_{k+1}, u_{k+1}). Each of the four problems can be solved in $O(n^2)$ time. Let us demonstrate this for problem (37), which computes $\hat{y}_k$. We first need to compute the objective vector Pu_k. This already requires $O(n^2)$ time. After this, the resulting minimization problem can be solved in only $O(n)$ time. This is easy to understand for (38) and (40); for (37) and (39) it can be realized by using the approach used in [3].

The MPBP is presented in Algorithm 1. In this algorithm $\tilde{v} = \tilde{y}^{\mathcal{L}^\perp} = \tilde{y} - P_A\tilde{y}$. As usual, we use an iteration counter k as subscript of all relevant vectors. The input is the matrix P_A and vectors $y \in \Delta$ and $u \in \mathcal{B}$. The MPBP is initialized with $\tilde{y} = y_1 = y$ and $\tilde{u} = u_1 = u$. At the end of the k-th iteration $\tilde{y}$ denotes the average of the iterates $\hat{y}_1$ to $\hat{y}_k$, and similar for $\tilde{u}$.

Algorithm 1: $[\tilde{y}, \tilde{u}, \mathcal{J}, case] = \text{MIRROR-PROX BP}(P_A, y, u)$

INITIALIZE: $k = 0$; $y_1 = y$; $\tilde{y} = y$; $u_1 = u$; $\tilde{u} = u$; $case = 0$; $\mathcal{J} = \emptyset$;
while $case = 0$ **do**
  $k = k + 1$
  if $P_A\tilde{u} > 0$ **then**
    $case = 1$ (problem 1 is feasible)
  else if $\sigma(\tilde{y} - P_A\tilde{y}) = 0$ **then**
    $case = 2$ (problem 1 is infeasible)
  else if $\sigma(\tilde{y} - P_A\tilde{y}) \le \frac{1}{2}$ **then**
    $case = 3$ (a proper cut has been found)
    find a nonempty index set $\mathcal{J}$ such that $\mathcal{J} \subseteq \left\{ j : \sigma_j(\tilde{v}) \le \frac{1}{2} \right\}$
  else
    $\hat{y}_k = \text{argmin}_{y\in\Delta} \left\{ \gamma y^T P_A u_k + \frac{1}{2} \left\| y - y_k \right\|^2 \right\}$
    $\hat{u}_k = \text{argmin}_{u\in\mathcal{B}} \left\{ -\gamma u^T P_A y_k + \frac{1}{2} \left\| u - u_k \right\|^2 \right\}$
    $y_{k+1} = \text{argmin}_{y\in\Delta} \left\{ \gamma y^T P_A \hat{u}_k + \frac{1}{2} \left\| y - y_k \right\|^2 \right\}$
    $u_{k+1} = \text{argmin}_{u\in\mathcal{B}} \left\{ -\gamma u^T P_A \hat{y}_k + \frac{1}{2} \left\| u - u_k \right\|^2 \right\}$
    $\tilde{y} = \frac{1}{k} \left((k-1)\tilde{y} + \hat{y}_k \right)$
    $\tilde{u} = \frac{1}{k} \left((k-1)\tilde{u} + \hat{u}_k \right)$

At the start of the MPBP we first check (in line 4) if $\tilde{u}$ is primal feasible with positive α. If so, we put $case = 1$, which means that a feasible solution of problem (1) has been found. Otherwise, we compute $\sigma(\tilde{v})$. If $\sigma(\tilde{v}) = 0$, problem (1) is infeasible, by Lemma 3.3. This is expressed by putting $case = 2$. Otherwise $\sigma(\tilde{v}) > 0$. If $\sigma(\tilde{v}) \leq 0.5$ then we have found a proper cutting vector, and we put $case = 3$. We then also determine all other proper cuts (in line 10).

In the remaining case we have $\sigma(\tilde{v}) > 0.5$, and then Nemirovski's Mirror-Prox method is used to compute new values for $\hat{y}_k$, $\hat{u}_k$, y_{k+1} and u_{k+1}. These values are the solutions of easy to solve minimization problems. Then $\tilde{y}$ is updated in such a way that it becomes the average of the iterates $\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_k$, and similarly for $\tilde{u}$. After this we enter the while loop again.

As we show in the next lemma, the BP stops after at most $2n\sqrt{n}$ iterations, and then $case \in \{1, 2, 3\}$. In each of these three cases the BP returns the value of the variable *case* and the current values of $\tilde{y}$, $\tilde{u}$ to the MA, as well as the set $\mathcal{J}$ of indices for which proper cuts has been found and the values of the upper bounds for the related entries in x.

Lemma 7.1. *With $\gamma = \frac{1}{2}$, the MPBP stops after at most $2n\sqrt{n}$ iterations.*

Proof: After the k-th iteration we have, by Theorem 6.5,

$$\left\|P\tilde{y}\right\| - \min(P\tilde{u}) \leq \frac{1}{k\gamma}, \quad k \geq 1.$$

Since $P\tilde{u}$ is not primal feasible, $\min(P\tilde{u}) \leq 0$. Hence we obtain

$$\left\|P\tilde{y}\right\| \leq \frac{1}{k\gamma}.$$

According to Lemma 3.4 the vector $\tilde{y}$ is cutting if $\left\|P\tilde{y}\right\| < \frac{1}{n\sqrt{n}}$. This certainly holds if

$$\frac{1}{k\gamma} < \frac{1}{n\sqrt{n}}.$$

Since $\gamma = \frac{1}{2}$, it follows that if $k \geq 2n\sqrt{n}$ the MPBP will have stopped. Hence the lemma follows. ■

Theorem 7.2. *Each execution of the MPBP needs at most $O(n^{3.5})$ arithmetic operations.*

Proof: We established before that each iteration of the MPBP requires $O(n^2)$ time. The number of iterations being at most $2n\sqrt{n}$, the theorem follows. ■

The above theorem makes clear that we have achieved an improvement over the original BP's in [4] and [12] which require $O(n^4)$ arithmetic operations.

8. Modified Main Algorithm

In order to solve (1) one needs to call the MPBP several times by another algorithm, named the Modified main algorithm (MMA), which is described in Algorithm 2. It is a simplified

version of the MMA in [12]. In [12] we used a property that is due to Khachiyan [7], namely that in the feasible case there exists a positive number κ such that positive entries in a basic feasible solution of (3) are larger than or equal to κ. The need for this number was a drawback, because it is not known in advance. In the new MMA we avoid the use of this number. We only use that (1) is infeasible if and only if the MPBP returns $case = 2$.

Algorithm 2: $[x, case] =$ MODIFIED MAIN ALGORITHM(A)

1: INITIALIZE: $d = \mathbf{1}$; $y = \mathbf{1}/n$; $u = 0$; $case = 3$;
2: **while** $case = 3$ **do**
3: $\quad P_A = I - A^T(AA^T)^{-1}A$ /* compute projection matrix */
4: $\quad [y, u, \mathcal{J}, case] =$ Mirror-Prox BP(P_A, y, u) /* call the MPBP */
5: $\quad$ **if** $case = 1$ **then**
6: $\quad\quad D = \text{diag}\,(d)$
7: $\quad\quad x = DP_Au$ /* x solves problem 1 */
8: $\quad$ **else if** $case = 2$ **then**
9: $\quad\quad x = 0$ /* problem 1 is infeasible */
10: $\quad$ **else**
11: $\quad\quad d_{\mathcal{J}} = d_{\mathcal{J}}/2$ /* update d, A, y, u */
12: $\quad\quad A_{\mathcal{J}} = A_{\mathcal{J}}/2$
13: $\quad\quad y_{\mathcal{J}} = y_{\mathcal{J}}/2$
14: $\quad\quad u_{\mathcal{J}} = u_{\mathcal{J}}/2$

The new MMA is initialized with $d = \mathbf{1}$ and $case = 3$, and with y and u as the centres of Δ of $\mathcal{B}$, respectively. At the start of the while loop the projection matrix P_A onto the null space of A is computed. Then the MPBP is called with P_A, y and u as input. The MPBP returns to the MMA with the quadruple $(y, u, \mathcal{J}, case)$ as output. If $case = 1$ the BP has found a u such that $x = DP_Au$ is feasible for problem (1); if $case = 2$ it halts with $x = 0$, indicating that (1) has no positive solution. Finally, if the MPBP returns $case = 3$, it has found proper cuts, indexed by the set $\mathcal{J}$. Then the entries in $d_{\mathcal{J}}$ are divided by 2 and we rescale A, y and u as indicated in lines 11-14. After this we enter the while loop again. Then the MPBP will be called again, and so on.

Theorem 8.1. *The execution of the MMA needs at most* $O(n^{3.5}\log_2(1/\delta_A))$ *time.*

Proof: From the Introduction we recall that the MMA calls the MPBP at most $\log_2(1/\delta_A)$ times. In each iteration the MMA needs $O(n^3)$ time for the computation of P_A before calling the MPBP. After this the MPBP needs $O(n^{3.5})$ time. So the time per MMA-iteration is $O(n^{3.5})$. Hence in total the MMA will require at most $O(n^{3.5}\log_2(1/\delta_A))$ time. ■

9. Numerical comparisons

Like our MMA, the so-called Enhanced Projection and Rescaling Algorithm (EPRA) in [9] is designed to solve problem (1). Just as our MMA uses the MPBP as BP, the EPRA uses the Smooth Perceptron Scheme (SPS) as BP. As shown in [9], the SPS is by far more efficient than the three other candidates in [9]: the Perceptron Scheme, the Von Neumann

Scheme and the Von Neumann Scheme with Away Steps. Therefore, following [9], we only considered SPS as a serious candidate for the BP in the EPRA.

Both the MPBP and the SPS establish feasibility of the primal or the dual problem, or they generate a 'proper' cut, where the meaning of 'proper' depends on a chosen threshold value. Below we denote this threshold value as τ. [3] Recall that in our MPBP (Algorithm 1) we took $\tau = \frac{1}{2}$, but any other number in the interval $(0, 1]$ would have been appropriate.

A nice feature in [9] is that the authors found a way to construct random matrixes A with a prescribed value of δ_A [9, Proposition 3.1]. This generator will be also used in our experiments.

As in [9] we start by a computational comparison of the two competing BP's in the next subsection, whereas in Subsection 9.2 we compare our MMA and the main algorithm EPRA in [9]. In both sections we use an appropriate variant of the Matlab code that was used for the experiments in [9] and that has been made publicly available at the website

http://www.andrew.cmu.edu/user/jfp/epra.html.

By using this code it was easy to make the desired comparisons. We simply ignored the lines related to the three BPs in [9] that differ from the SPS and replaced them by the MPBP for the experiments in Section 9.1. A similar approach made it relatively easy to set up the comparison in Section 9.2.

One more remark should be made. Though the MPBP as presented in Algorithm 1 gave CPU times that were competing with those of the SPS, significantly better results were obtained by modifying lines 16–17. Recall that line 16 forces $\tilde{y}$ to be the average of the iterates $\hat{y}_1, \hat{y}_2 \ldots, \hat{y}_n$. Instead of this we simply used $\tilde{y} = y_{k+1}$. Similarly, in our implementation we replaced the command in line 17 by $\tilde{u} = u_{k+1}$.

9.1. Comparison of MPBP and SPS

As in [9], for the comparison of MPBP and SPS we used six tables. Besides the parameters τ and δ_A we use N to denote the number of instances per size, and as in [9], an iteration limit, which we denote as K. Per table these parameters are as given in Table 2. Except for the values of τ in Tables 3 and 4, the values are the same as in [9]. These two tables served in [9] to demonstrate the effect of the iteration limit K. In the case where K is finite the limit is active only in Tables 5 and 6. Therefore, we used in Tables 3 and 4 the same parameters as in Tables 5 and 6, respectively, but without iteration limit.

In each of the six tables below we considered matrices A of four different sizes, as indicated in the first two columns of each table. For each size N different matrices A were generated.

Table 2. Parameter values in the six tables.

Table	τ	δ_A	K	N
3	0.0001	1	∞	100
4	0.0001	0.001	∞	100
5	0.0001	1	10,000	1000
6	0.0001	0.001	10,000	1000
7	0.1	1	10,000	1000
8	0.1	0.001	10,000	1000

Table 3. Naive random instances ($\tau = 0.0001, \delta_A = 1, N = 1000, K = \infty$).

size(A)		av. iterations		av. cputime		success rate	
m	*n*	SPS	MPBP	SPS	MPBP	SPS	MPBP
100	200	254.33	73.65	0.0210	0.0052	1.000	1.000
250	500	663.76	138.60	0.1421	0.0303	1.000	1.000
500	1000	924.11	201.59	1.2723	0.3059	1.000	1.000
1000	2000	885.26	219.27	4.8760	1.6163	1.000	1.000

Table 4. Controlled condition instances ($\tau = 0.0001, \delta_A = 0.001, N = 100, K = \infty$).

size(A)		av. iterations		av. cputime		success rate	
m	*n*	SPS	MPBP	SPS	MPBP	SPS	MPBP
100	1000	14,489.95	2,563.11	19.0735	3.8679	1.000	1.000
200	1000	14,462.39	1,929.51	18.9831	2.8488	1.000	1.000
800	1000	8,972.84	1,318.23	12.8103	2.1570	1.000	1.000
900	1000	7,542.89	1,950.49	10.3521	3.0315	1.000	1.000

Table 5. Naive random instances ($\tau = 0.0001, \delta_A = 1, N = 1000, K = 10,000$).

size(A)		av. iterations		av. cputime		success rate	
m	*n*	SPS	MPBP	SPS	MPBP	SPS	MPBP
100	200	208.59	60.34	0.0171	0.0043	1.000	1.000
250	500	425.15	118.78	0.0969	0.0271	0.998	0.999
500	1000	579.90	145.78	0.8463	0.2428	0.997	1.000
1000	2000	823.48	208.45	4.8652	1.6833	0.989	1.000

Table 6. Controlled condition instances ($\tau = 0.0001, \delta_A = 0.001, N = 1000, K = 10,000$).

size(A)		av. iterations		av. cputime		success rate	
m	*n*	SPS	MPBP	SPS	MPBP	SPS	MPBP
100	1000	9550.41	2654.55	13.5566	4.3351	0.662	0.956
200	1000	9567.92	1773.15	13.5134	2.8833	0.707	0.977
800	1000	8646.16	1276.30	11.7611	1.9800	0.999	1.000
900	1000	7578.60	1904.01	10.4732	2.9954	1.000	1.000

After the computation of the projection matrix P_A both the SPS and the MPBP were called, with input P_A, the centre u of $\mathcal{B}$ and the centre y of Δ. The tables show the averages of, respectively, the number of iterations and the CPU time in seconds. The last two columns show the 'rate of success', which is the ratio of successful runs for the specific problem size; a run is considered successful if the iteration limit K was not the reason for stopping the run. Thus we see that in Tables 3, 4, 7 and 8 all runs were successful. But in Table 5, line 2, 2 runs of the SPS and 1 run of the MPBP were not successfull and in Table 6 many runs failed to establish feasibility or infeasibility, or to generate a cut. Obviously, low values of the success rate are due the small value of τ in Tables 5 and 6. Note that in the extreme case that we would put $\tau = 0$, any BP would stop its execution only after having solved the problem, i.e. only after having found a solution or having found proof for infeasibility of (1).

The six tables in this section permit us to conclude that in most cases MPBP is more efficient than SPS. In only two cases (in Table 8) SPS behaves slightly better than MPBP,

Table 7. Naive random instances ($\tau = 0.1, \delta_A = 1, N = 1000, K = 10,000$).

size(A)		av. iterations		av. cputime		success rate	
m	*n*	SPS	MPBP	SPS	MPBP	SPS	MPBP
100	200	66.09	31.73	0.0060	0.0024	1.000	1.000
250	500	115.28	53.82	0.0285	0.0129	1.000	1.000
500	1000	174.84	77.01	0.2557	0.1252	1.000	1.000
1000	2000	254.22	108.07	1.5426	0.8847	1.000	1.000

Table 8. Controlled condition instances ($\tau = 0.1, \delta_A = 0.001, N = 1000, K = 10,000$).

size(A)		av. iterations		av. cputime		success rate	
m	*n*	SPS	MPBP	SPS	MPBP	SPS	MPBP
100	1000	126.86	160.18	0.2140	0.3046	1.000	1.000
200	1000	140.21	158.28	0.2224	0.2820	1.000	1.000
800	1000	218.91	24.98	0.3720	0.0488	1.000	1.000
900	1000	199.88	19.58	0.2801	0.0313	1.000	1.000

Table 9. Parameter settings for Gurobi (version 9.0.3) and Mosek (version 9.1.0).

Mosek	Gurobi
param.MSK_IPAR_PRESOLVE_USE = 'MSK_OFF'	params.presolve = 0
param.MSK_IPAR_INTPNT_BASIS = 'MSK_BI_NEVER'	params.crossover = 0
param.MSK_DPAR_INTPNT_TOL_INFEAS = 1e-14	params.barconvtol = 1e-14

namely when $\delta_A = 0.001$ and $m \ll n$. It is an interesting question whether this can be explained.

9.2. Comparison of MMA with EPRA, Gurobi and Mosek

In this section we compare our Main Algorithm (MMA) with the Enhanced Projection and Rescaling Algorithm (EPRA) in [9] and also with the commercial packages Gurobi and Mosek. For that purpose we used the publicly available Matlab code that was used to generate Table 7 in [9]; this table contains for eight different sizes the averages of the number of iterations of the EPRA, the number of iterations of SPS and the CPU time for N randomly generated matrices A. We modified this code in such a way that every matrix A was also sent to our MMA and to Gurobi and Mosek. For Gurobi and Mosek we used the parameter setting as given in Table 9. Tables 10 and 11 show the results.

Table 10. MMA versus EPRA, Gurobi and Mosek ($N = 500, \tau = 0.5, x > 0, \delta = 0.001$).

size(A)		av. iterations		av. iterations		average cputime				fract.
m	n	EPRA	MMA	SPS	MPBP	EPRA	MMA	Gurobi	Mosek	feas.
100	200	10.39	11.73	725.20	97.92	0.0484	0.0148	0.0130	**0.0080**	1.0000
250	500	12.11	13.07	1452.15	196.04	0.3901	0.1141	0.1149	**0.0568**	1.0000
100	1000	9.16	9.15	1842.24	235.96	2.0005	0.4595	0.0650	**0.0392**	1.0000
200	1000	9.97	9.86	1996.34	169.71	2.1449	0.3925	0.1108	**0.0821**	1.0000
500	1000	12.99	13.14	2504.04	317.24	2.4293	0.7992	0.2950	**0.2190**	1.0000
800	1000	13.94	13.73	4000.80	484.64	3.6971	1.4788	0.7066	**0.4504**	1.0000
900	1000	12.60	13.31	4154.81	548.44	3.7366	1.7188	0.9319	**0.5308**	1.0000
1000	2000	13.16	12.82	4349.33	481.47	21.8629	5.8430	1.6831	**1.1653**	1.0000

Table 11. MMA versus EPRA, Gurobi and Mosek ($N = 1000$, $\tau = 0.5$, x free).

size(A)		av. iterations		av. iterations		average cputime				fract.
m	n	EPRA	MMA	SPS	MPBP	EPRA	MMA	Gurobi	Mosek	feas.
100	200	1.41	4.48	146.86	35.45	0.0088	**0.0058**	0.0134	0.0065	0.5168
250	500	1.32	4.57	264.94	58.35	0.0499	**0.0395**	0.1159	0.0426	0.4760
100	1000	1.00	1.00	5.50	3.31	0.0692	**0.0170**	0.0531	0.0257	1.0000
200	1000	1.00	1.00	9.70	4.45	0.0632	**0.0208**	0.0734	0.0532	1.0000
500	1000	1.21	4.54	400.10	89.16	0.3211	0.3028	0.2862	**0.2012**	0.4885
800	1000	1.00	1.02	9.78	3.00	0.0661	**0.0538**	0.5432	0.3029	0.0000
900	1000	1.00	1.69	5.50	1.69	**0.0681**	0.1085	0.6406	0.3383	0.0000
1000	2000	1.19	4.88	638.81	137.98	2.3357	1.9746	2.0024	**0.9475**	0.5020

In Table 10 the matrices A are generated by the matrix generator in the EPRA code. This implies that there exists a positive vector in the null space of A. In other words, problem (1) is always feasible in Table 10. Table 11 differs only in the way the matrices A are generated; we allow the resulting problem (1) to be infeasible, by simply using the Matlab command `A = rand(m,n)-0.5`. This difference is indicated by writing '$x > 0$' in the caption of Table 10 and 'x free' in the caption of Table 11.

The first two columns in both tables contain the eight sizes that were considered; these are the same as in [9]. The next two columns give the average numbers of iterations of the EPRA and the MMA, respectively, whereas the subsequent two columns give the averages of the total number of iterations of the SPS and the MPBP, respectively. Then the next four columns give the average CPU times (in seconds) for both cases and also for Gurobi and for Mosek. Finally, the last column presents the percentage of the N instances that turned out feasible. By the construction of the matrices A, this fraction is always 1 in Table 10.

For Gurobi and Mosek we used the parameter setting as given in Table 9.

We finally discuss the values of the parameters τ, δ_A and N. The matrix generator in [9] has the property that the matrix A is always such that δ_A is given and positive. This implies that problem (1) is always feasible, as stated before. In practice, deciding on feasibility or infeasibility of a given problem is one of the main tasks of a solution method. In the infeasible case we have $\delta_A = 0$. In order to allow this important case in our experiments, we also included Table 11. By their construction, on average the matrices that were generated had an equal division of the signs of the entries. It may therefore be not surprising that if $n \approx 2m$ this leads–roughly spoken–to an equal division of feasible and infeasible instances, as we see in the lines 1, 2, 5 and 8 of the table. The other lines give rise to an interesting probabilistic question: should we have expected that each of the 1000 instances were feasible if $n \geq 5m$ and infeasible if $n \leq 1.25m$?

10. Conclusions

The results in Section 9.1 reveal that the Mirror-Prox Basic Procedure (MPBP) that we introduced in this paper competes very well with the Smooth Perceptron Scheme (SPS) of [9]. The comparison in Section 9.2 confirms these results for the numerical behaviour of the Enhanced Projection and Rescaling Algorithm (EPRA), which uses the SPS as a subroutine, and the Modified Main Algorithm (MMA), which uses the MPBP as subroutine. The barrier codes for linear optimization in the commercial packages Gurobi and Mosek do very well in this comparison. It should be mentioned that this became true only

after changing the default parameter setting as indicated in Tabel 9. In all considered cases Mosek beats Gurobi, but it is quite surprising that the straightforward implementations of EPRA and MMA compete very well with both. In six of the eight cases MMA is faster than Mosek.

Also a new challenge arises, because the implementation of the MMA that we used differs from the method that we analysed theoretically. Can one prove that the implemented method also requires only polynomial time?

Two other points are worth to be noticed. First, in the MMA, after the first time, the MPBP is initialized with the vectors y and u generated in the previous iteration of the MMA, hence not starting every time from $y = \mathbf{1}/n$ and $u = 0$. Second, Table 11 suggests that when generating random matrices, the value of n/m strongly affects the feasibility of the problem.

Notes

1. As we showed in [12], system (8) is feasible if and only if $\|v\|^2 \leq \max(v)$. Since $\max(v) \leq \|v\|$ this only holds if $\|v\| \leq 1$. It may be worth mentioning that this result is not used in this paper.
2. At this stage the mistake in [12] becomes visible. There we incorrectly assumed that $q = p$, or, in other words, that v_I consists of the positive entries in v. This, however, yields a dual feasible solution that may be not optimal.
3. In [9] the related parameter is denoted as ϵ.
4. The original text, in Dutch: Ich thu dat meine, Soo viel mijr God bescheert, Ein ander thu dat seine, Soo wirdt de Const ghemheert.

Acknowledgments

Thanks are due to the anonymous referees for their comments. Their recommendation to use the Matlab code in [9] for the computational comparisons were very useful. Thanks are also due to Erling Andersen (Mosek) who suggested the parameter settings in Table 9 for Mosek; the parameters for Gurobi were set accordingly. Citing the Dutch mathematician Ludolph van Ceulen (1540–1610)–who became famous for his computation of 36 decimals of the number π–we offer this paper to the readers:[4]

We did what we could, as much as God enabled us to do. Let everyone do his sake, in this way science blossoms.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Notes on contributors

Zhang Wei (1986) is an Assistant Professor in the School of Mathematics, South China University of Technology. He received his bachelor's degree at Peking University (2010) and PhD's degree in the University of Chinese Academy of Sciences (2015). After that he worked in Shanghai University and the National University of Singapore as a postdoc research fellow. His research interest lies in the first order methods in mathematical optimization and spectral graph theory.

Kees Roos (1941) has hold a chair on Optimization Technology at Delft University of Technology until 2006, when he retired. From 1998 to 2002 he was a part-time professor at Leiden University. The past 25 years his research concentrated on interior-point methods for linear and convex optimization. He is a (co-)author of several books and more than hundred papers in refereed journals.

He is (or was) member of the editorial board of several journals, among them the SIAM Journal on Optimization. He was secretary/treasurer of the SIAM Activity Group on Optimization. He supervised a large number of research projects, among them the Dutch nationwide NWO-project High Performance Methods for Mathematical Optimization, and the project Discrete Mathematics and Optimization of the (Dutch) Stieltjes Institute. He was also involved in a big project 'Optimal safety of the dikes in the Netherlands', financed by the Dutch government. that received the Franz Edelman award 2013 of INFORMS. He shared the 2011 Khachiyan prize of the INFORMS Optimization Society with Jean-Philippe Vial (University of Geneva).

References

[1] E.D. Andersen and K.D. Andersen, *The MOSEK interior-point optimizer for Linear Programming: an implementation of the homogeneous algorithm*, in *High Performance Optimization Techniques*, S. Zhang, H. Frenk, C. Roos, and T. Terlaky, eds., Kluwer Academic Publishers, Dordrecht, The Netherlands, 1999, pp. 197–232.

[2] A. Ben-Tal and A. Nemirovski, *Lectures on modern convex optimization*, *analysis, algorithms, and engineering applications*, MPS/SIAM Series on Optimization, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2001.

[3] P. Brucker, *An $O(n)$ algorithm for quadratic knapsack problems*, Oper. Res. Lett. 3(3) (1984), pp. 163–166.

[4] S. Chubanov, *A polynomial projection algorithm for linear feasibility problems*, Math. Program. 153 (2015), pp. 687–713. DOI: 10.1007/s10107-014-0823-8.

[5] 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*, Tech. Rep. SOL 92-5, Systems Optimization Laboratory. Department of Operations Research. Stanford University, Stanford, USA, Oct 1992.

[6] N. He, *IE 598 Big Data Optimization*, Lecture Notes, Vol. 18, University Of Illinois At Urbana-Champaign, Fall, Oct 2016.

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

[8] A. Nemirovski, *Prox-method with rate of convergence $O(1/t)$ for variational inequalities with Lipschitz continuous monotone operators and smooth convex-concave saddle point problems*, SIAM J. Optim.15(1) (2004), pp. 229–251.

[9] J. Peña and N. Soheili, *Computational performance of a projection and rescaling algorithm*, Optim. Methods Softw. (2019), pp. 1–18. https://www.tandfonline.com/doi/full/10.1080/10556788.2019.1615910.

[10] C. Roos, *Linear optimization: Theorems of the alternative*, in *Encyclopedia of Optimization*, C.A. Floudas and P.M. Pardalos, eds., Vol. III, Kluwer Academic Publishers, Dordrecht, The Netherlands, 2001, pp. 181–184, ISBN 0-19-512594-0.

[11] C. Roos, *On Chubanovs method for solving a homogeneous inequality system*, in *Numerical Analysis and Optimization. NAO-III, Muscat, Oman, January 2014*, M. Al-Baali, L. Grandinetti, and A, Purnama, eds., Springer Proceedings in Mathematics & Statistics, Switzerland, 2015, pp. 319–338, ISBN 978-3-319-17688-8.

[12] C. Roos, *An improved version of Chubanov's method for solving a homogeneous feasibility problem*, Optim. Methods Softw. 33(1) (2018), pp. 26–44. DOI: 10.1080/10556788.2017.1368509.

[13] E. Stiemke, *Über positive Lösungen homogener linearer Gleichungen*, Mathematische Annalen76 (1915), pp. 340–342.

[14] Available at https://www.gurobi.com/.

[15] Available at https://www.mosek.com/.