A polynomial projection algorithm for linear programming

Sergei Chubanov

Institute of Information Systems at the University of Siegen, Germany

e-mail: sergei.chubanov@uni-siegen.de

May 3, 2013

Abstract

We propose a polynomial algorithm for linear programming. The algorithm represents a linear optimization or decision problem in the form of a system of linear equations and non-negativity constraints. Then it uses a procedure that either finds a solution for the respective homogeneous system or provides the information based on which the algorithm rescales the homogeneous system so that its feasible solutions in the unit cube get closer to the vector of all ones. In a polynomial number of calls to the procedure the algorithm either proves that the original system is infeasible or finds a solution in the relative interior of the feasible set.

1 Introduction

In this paper we present a polynomial algorithm for linear programming. As compared to the earlier paper [2], we achieve a considerable reduction of the running time. The

algorithm is based on a procedure whose input is a homogeneous system of linear inequalities. This procedure, that we will call the basic procedure, either finds a solution with strictly positive components or provides the main algorithm with the information based on which the algorithm decides how to rescale the system so that the feasible solutions in the unit cube get closer to the vector of all ones. The basic procedure is applied to a system $Px > \mathbf{0}$ where P is a projection matrix associated with the original system. The procedure constructs a sequence of points until a stopping condition is met. To construct the next point, the procedure finds a constraint $p^T x > 0$ violated by the current point $x^{(s)}$. The next point has the form $x^{(s+1)} = \alpha x^{(s)} + (1-\alpha)p$, where the coefficient α is chosen so that $x^{(s+1)}$ is the projection of a zero vector on the segment $[x^{(s)}, p]$. We can see some similarity between the basic procedure and the relaxation method for linear inequalities of Agmon [1] and Motzkin and Schoenberg [9]. The relaxation method, in the same situation, would construct a point $x^{(s+1)} = x^{(s)} + \gamma p$, where γ is some parameter. (For a further reference see, for instance, Goffin [5], [6].)

In the subsequent sections we first develop a polynomial algorithm for homogeneous systems and then apply the algorithm to systems of different types. In particular, we show that if a linear system $Ax = b, x \geq \mathbf{0}$, has a solution $x > \mathbf{0}$ of size S in the unit cube then the algorithm solves the system in $O(n^3 S)$ time. We also show that the algorithm can be used to find a solution in the relative interior of the feasible set in $O(n^4 L)$ time where L is the binary size of the system. The algorithm also finds a linear system defining the affine hull of the feasible set. A variant of the algorithm either finds a solution of a system $Ax = b, \mathbf{0} \leq x \leq \mathbf{1}$, or proves that the system has no 0-1 solutions in strongly polynomial time $O(n^4)$. This estimate of the running time improves that of our previous algorithm [2] by a factor of n^2.

2 Algorithm for homogeneous linear systems

Let $\mathbf{0}$ denote a vector of all zeros and $\mathbf{1}$ denote a vector of all ones. The dimensions of these vectors, i.e., their numbers of components and whether they are columns or rows, will be

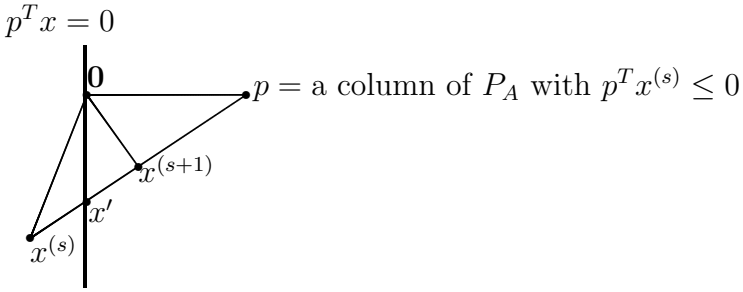

Figure 1: One iteration of the basic procedure

determined by the context. The unit row vector whose ith component equals 1 is denoted by $\mathbf{e}_i$.

We consider a homogeneous system

$$Ax = \mathbf{0}, x \geq \mathbf{0},$$

where A is an $m \times n$ rational matrix of rank m. The system defines a cone in $\mathbb{R}^n$.

In this section we describe an algorithm that either finds a positive feasible solution, i.e., a feasible solution x with $x > \mathbf{0}$ (the condition $x > \mathbf{0}$ requires all the components of x to be strictly positive), or proves that there are no such solutions. Its running time, by which we understand the number of arithmetic and the other elementary operations performed, is polynomially bounded in the binary size of the system.

2.1 Basic procedure

Let P_A denote the matrix of the orthogonal projection onto the subspace $\{x|Ax = \mathbf{0}\}$ (in other words, onto the null space of A). That is,

$$P_A = I - A^T(AA^T)^{-1}A.$$

(I is the identity matrix.)

It is convenient to consider the associated system $P_A x > \mathbf{0}$. If x^* is its feasible solution, then $P_A x^*$ is a positive feasible solution of the original system because $AP_A x^* = \mathbf{0}$. If x^* is in addition a linear combination of the columns of P_A, then $P_A x^* = x^*$.

The following argument is illustrated in Figure 1. Let $x^{(s)}$ be a non-zero point in $\mathbb{R}^n$ that

violates some inequality $p^T x > 0$ of the system $P_A x > \mathbf{0}$, i.e., we have

$$p^T x^{(s)} \leq 0.$$

As P_A is symmetric, p is the respective column of P_A. Consider the point

$$x^{(s+1)} = \alpha x^{(s)} + (1-\alpha)p \tag{1}$$

where α is chosen so that $x^{(s+1)}$ is the projection of $\mathbf{0}$ on the segment $[x^{(s)}, p]$, which means that α is computed as

$$\alpha = \frac{p^T(p - x^{(s)})^T}{\|p - x^{(s)}\|^2}.$$

The α belongs to $[0, 1]$. (This can be verified by using $p^T x^{(s)} \leq 0$.) Note that

$$(p - x^{(s)})^T x^{(s+1)} = 0.$$

So $x^{(s+1)}$ is at the same time the projection of $\mathbf{0}$ on the line generated by the segment $[x^{(s)}, p]$. In other words, $x^{(s+1)}$ is the altitude of the right triangle formed by the points p, $\mathbf{0}$, and x' where x' is the intersection point of $[x^{(s)}, p]$ with the hyperplane $\{x | p^T x = \mathbf{0}\}$. (The existence of the intersection point follows from $p^T x^{(s)} \leq 0$.)

The main ingredient of our algorithm is a procedure that constructs a sequence of non-zero points connected with each other by the equation (1). We call this procedure the *basic procedure*. It starts with a non-zero point $x^{(0)}$ being a convex combination of the columns of P_A. This implies that all points in the sequence are convex combinations of the columns of P_A. That is, each point $x^{(s)}$ can be represented in the form

$$x^{(s)} = P_A (y^{(s)})^T,$$

where $y^{(s)}$ is a nonnegative row vector with $y^{(s)}\mathbf{1} = 1$. Note that

$$P_A x^{(s)} = x^{(s)}.$$

If $x^{(s)} > \mathbf{0}$, then the basic procedure returns $x^{(s)}$ because $x^{(s)}$ is a positive feasible solution of the homogeneous system. Otherwise, the procedure finds an index i with $x_i^{(s)} \leq 0$. Then

$p^T x^{(s)} \leq 0$ where p^T is the ith row of P_A. The p is at the same time the ith column because P_A is symmetric. The next point in the sequence can be represented as

$$x^{(s+1)} = P_A (y^{(s+1)})^T,$$

where

$$y^{(s+1)} = \alpha y^{(s)} + (1 - \alpha)\mathbf{e}_i.$$

As we will see, each iteration of the basic procedure increases the value $\frac{1}{\|x^{(s)}\|}$ associated with the current point. We call such values *potentials*. One of the stopping conditions that we develop later is reached if the potential is sufficiently large.

Let us first consider the degenerate case where $\mathbf{0}$ belongs to $[x^{(s)}, p]$. In this case $\mathbf{0} = \alpha x^{(s)} + (1 - \alpha)p$, which implies that $\mathbf{0} = \alpha P_A (y^{(s)})^T + (1 - \alpha)p$. Then $y = \alpha y^{(s)} + (1 - \alpha)\mathbf{e}_i$ satisfies $yP_A = \mathbf{0}$. Let x^* be a solution of the homogeneous system. Since $P_A x^* = x^*$, it follows that $yx^* = 0$. Then $x^*_k = 0$ for all k with $y_k > 0$. In particular, $y_i > 0$ because $\alpha < 1$ due to $x^{(s)} \neq \mathbf{0}$. It follows that the homogeneous system has no positive solutions in the degenerate case, and, moreover, that we can produce a nonempty subset of indices k such that $x^*_k = 0$ for each feasible solution x^*.

One can also note that in the degenerate case y has the form $y = zA$, where $z = yA^T(AA^T)^{-1}$. That is, we could apply a variant of Farkas' lemma such as the one by Stiemke [11] that says that the homogeneous system has no positive solutions if and only if the system $y = zA, y \geq \mathbf{0}, y \neq \mathbf{0}$, is feasible.

The lemma below is a direct consequence of the Pythagorean theorem. Actually, it determines a well-known relation between the altitude from the right angle and the legs of the respective right triangle.

We use the same notation as before. Recall that x' is the intersection of $[x^{(s)}, p]$ with the hyperplane $\{x | p^T x = 0\}$.

Lemma 2.1 *If* $\mathbf{0}$ *is not in* $[x^{(s)}, p]$, *then*

$$\frac{1}{\|x^{(s+1)}\|^2} = \frac{1}{\|x'\|^2} + \frac{1}{\|p\|^2}.$$

Proof. It is sufficient to consider the special case with $\|p - x'\| = 1$. The Pythagorean theorem implies that

$$1 = \|p - x'\|^2 = \|x'\|^2 + \|p\|^2.$$

Let $x^{(s+1)}$ be expressed as $x^{(s+1)} = x' + \lambda(p - x')$. As $x^{(s+1)}$ is the projection of $\mathbf{0}$ on the line generated by $[x^{(s)}, p]$, we have

$$0 = (x^{(s+1)})^T(p - x') = -\|x'\|^2 + \lambda,$$

using the fact that $p^T x' = 0$. Then

$$\|x^{(s+1)}\|^2 = \|x'\|^2 + \lambda^2\|p - x'\|^2 - 2\lambda\|x'\|^2 = \lambda - \lambda^2 = \|x'\|^2(1 - \|x'\|^2) = \|x'\|^2\|p\|^2.$$

We can write

$$\frac{1}{\|x^{(s+1)}\|^2} = \frac{1}{\|x'\|^2\|p\|^2} = \frac{\|p - x'\|^2}{\|x'\|^2\|p\|^2} = \frac{\|x'\|^2 + \|p\|^2}{\|x'\|^2\|p\|^2} = \frac{1}{\|x'\|^2} + \frac{1}{\|p\|^2}.$$

■

If $\mathbf{0}$ does not belong to $[x^{(s)}, p]$, then, noting that $\|p\| \leq 1$ and that $\|x'\| \leq \|x^{(s)}\|$ because x' belongs to $[x^{(s)}, x^{(s+1)}]$, by the above lemma we conclude that

$$\frac{1}{\|x^{(s+1)}\|^2} = \frac{1}{\|x'\|^2} + \frac{1}{\|p\|^2} \geq \frac{1}{\|x^{(s)}\|^2} + 1.$$

In other words, the iteration of the basic procedure increases the square of the potential by at least one in the above case.

The objective of the basic procedure is to find at least one of the following objects:

(i) An index k such that $x_k^* \leq \frac{1}{2}$ for all feasible solutions x^* in $[0, 1]^n$;

(ii) An index k such that $x_k^* = 0$ for all feasible solutions x^*; or

(iii) A feasible solution $x^* > \mathbf{0}$.

Now we will develop stopping conditions that guarantee one of the above results. If the current point $x^{(s)}$ is positive, we have (iii). If $x_i^{(s)} \leq 0$ and $\mathbf{0}$ is contained in $[x^{(s)}, p]$ where p is the ith column of P_A, then the index $k = i$ yields (ii). It remains to develop a way to

obtain (i). As P_A is symmetric, we can rewrite $x^{(s)}$ as $x^{(s)} = (y^{(s)}P_A)^T$. Let x^* be a solution of $Ax = \mathbf{0}$ in the unit cube $[0,1]^n$. (It follows that $\|x^*\| \leq \sqrt{n}$.) Then $P_A x^* = x^*$. Consider $y_k^{(s)} = \max y^{(s)}$. This component is positive. Using the Cauchy-Schwartz inequality, and the non-negativity of x^*, we write

$$y_k^{(s)} x_k^* \leq y^{(s)} x^* = y^{(s)} P_A x^* = (x^{(s)})^T x^* \leq \|x^{(s)}\| \|x^*\| \leq \|x^{(s)}\| \sqrt{n}.$$

It follows that

$$x_k^* \leq \frac{\|x^{(s)}\| \sqrt{n}}{y_k^{(s)}}.$$

Then the condition

$$y_k^{(s)} \geq 2\|x^{(s)}\| \sqrt{n} \tag{2}$$

guarantees that we have (i).

Another condition that implies (i) follows from the analysis of some dual problems. Let $y_k^{(s)} = \max y^{(s)}$. Consider the optimization problem

$$\begin{array}{ll} \max & x_k \\ \text{s.t.} & Ax = \mathbf{0}, \\ & \mathbf{0} \leq x \leq \mathbf{1}. \end{array}$$

The dual problem is written as

$$\begin{array}{ll} \min & w\mathbf{1} \\ \text{s.t.} & zA + w \geq \mathbf{e}_k, \\ & w \geq \mathbf{0}. \end{array}$$

Consider the row vector

$$z^0 = \frac{y^{(s)} A^T (AA^T)^{-1}}{y_k^{(s)}}.$$

Note that

$$z^0 A = \frac{y^{(s)} (I - P_A)}{y_k^{(s)}}.$$

Let $[\cdot]^+$ denote the replacement of the negative components by zeros. Choosing

$$w^0 = \frac{[y^{(s)} P_A]^+}{y_k^{(s)}} = \frac{[(x^{(s)})^T]^+}{y_k^{(s)}},$$

we ensure that (z^0, w^0) is feasible for the dual problem. If $w^0\mathbf{1} \leq \frac{1}{2}$, then $x^*_k \leq \frac{1}{2}$ for all feasible solutions x^* of the primal problem. This yields (i).

The condition $w^0\mathbf{1} \leq \frac{1}{2}$ can be written in the form

$$\max y^{(s)} \geq 2 \cdot [(x^{(s)})^T]^+\mathbf{1}. \tag{3}$$

Since

$$[(x^{(s)})^T]^+\mathbf{1} \leq \|x^{(s)}\|\sqrt{n},$$

it follows that (2) implies (3). Therefore the condition (3) dominates (2) in the sense that (3) allows us to find (i) no later than (2) does, in the course of the basic procedure.

2.2 Formal description and analysis of the basic procedure

In the main algorithm that we discuss later on, the initial vector $y^{(0)}$ for each call to the basic procedure uses the results of the previous call. For this technical purpose, the basic procedure can additionally return a non-zero row vector y^{out} which is considered at the *last but one* iteration, provided that more than one iteration is performed and the result is (i). This allows the basic procedure to start with a reasonably "good" point.

The procedure uses (3) as a stopping condition that implies (i). Our previous argument is summarized in the formal description of the procedure:

Procedure 2.1 BASIC PROCEDURE

Input: A matrix A and a row vector $y^{in} \geq \mathbf{0}$ with $y^{in}\mathbf{1} = 1$.
Output: (i), (ii), or (iii), and a row vector y^{out} in case (i).
$y^{(0)} := y^{in}$;
Construct P_A and $y^{(0)}P_A$;
$x^{(0)} := P_A(y^{(0)})^T$;
$s := 0$;
while $\max y^{(s)} < 2 \cdot [(x^{(s)})^T]^+\mathbf{1}$
    if $x^{(s)} > \mathbf{0}$ **then** return $x^{(s)}$;

else find i with $x_i^{(s)} \le 0$;

Let p be the ith column of P_A;

if $\mathbf{0}$ belongs to $[x^{(s)}, p]$ **then** set $k := i$ and return (ii);

Compute α;

$y^{(s+1)} := \alpha y^{(s)} + (1-\alpha)\mathbf{e}_i$;

$x^{(s+1)} := \alpha x^{(s)} + (1-\alpha)p$;

$s := s + 1$;

end

Find k with $y_k^{(s)} = \max y^{(s)}$;

if $s \ge 1$ **then** return (i) and $y^{out} = y^{(s-1)}$ **else** return (i) and $y^{out} = \mathbf{0}$.

Remark. By an iteration of the basic procedure we understand an iteration of the while-loop. Even if the loop terminates immediately after checking the condition, this is considered as an iteration. In this case $s = 0$ and the procedure sets the output vector y^{out} to $\mathbf{0}$. The matrix P_A can be constructed by a variant of Gaussian elimination in $O(n^3)$ arithmetic operations on the numbers whose sizes are bounded by a polynomial in size(A). (See Edmonds [3] and Renegar [10].)

The fact that the basic procedure can result only in (i), (ii), or (iii) means that in a polynomial number of calls to it one can either find a positive feasible solution or prove that no such solutions exist. Although this conclusion is evident, we will discuss some details because we need them later. If the procedure results in (ii), then there are no positive solutions. If (iii) is found, we return a positive solution. If (i) is found, then we replace the original system by the system

$$A'x = \mathbf{0}, x \ge \mathbf{0},$$

where A' is obtained from A by the division of the kth column by 2. A solution x^* of the original system is thereby uniquely mapped to the solution $x^* + x_k^* \mathbf{e}_k^T$ of the modified system. This mapping maps every feasible solution x^* in the cube $[0,1]^n$ to a solution in the same cube because the inequality $x_k^* \le 1/2$ holds by (i). So one can perform a sequence of calls to the basic procedure followed by the respective transformations of A, each transformation

being interpreted as the multiplication by 2 of the respective kth components of all solutions of the previous system. All solutions in the unit cube are mapped to solutions in *the same cube* during this process. Note that the transformations of A lead to a matrix $A'' = AM$, where M is a diagonal matrix whose entries on the main diagonal are *non-positive* powers of 2. The current system has the form

$$A''x = \mathbf{0}, x \geq \mathbf{0}.$$

So after a certain number of calls to the basic procedure, say, after T calls resulting in (i), we have a proof that the components of every solution x^* in $[0,1]^n$ must satisfy $x_i^* \leq 1/2^{t_i}$, $i = 1, \ldots, n$, where the sum of the nonnegative integers t_i is equal to T. If $t_i \geq const \cdot \text{size}(A)$, where size denotes the binary size and a constant *const* can be given explicitly, then well-known facts on the sizes of subdeterminants imply that $x_i^* = 0$ for all feasible solutions x^*. Thus, in $O(n \cdot \text{size}(A))$ calls to the basic procedure we either find a positive solution or prove that the system has no positive solutions. Of course, this bound can be too large and might not fully reflect our knowledge about the system. We let $L_{\min}$ further denote a known polynomial upper bound on the number of calls, followed by the respective transformations of A, such that exceeding the limit of $L_{\min}$ calls guarantees that the system has no positive solutions. For instance, $L_{\min}$ can be chosen so that, under the assumption that there is a positive solution, $1/2^{L_{\min}}$ would represent a lower bound on the smallest volume V of a box $[0, v_1] \times \ldots \times [0, v_n]$ which contains all feasible solutions in $[0,1]^n$. If a positive solution exists, then the choice of $L_{\min}$ does not influence the number of calls to the basic procedure because $V > 0$ in this case and a positive feasible solution will anyway be found in $O(\log \frac{1}{V})$ calls. If, for instance, for each index i there is a solution x^* in $[0,1]^n$ with $x_i^* \geq 1/2$, then $V \geq 1/2^n$ and a positive solution will be found in $O(n)$ calls. Similarly, we can show that $\log \frac{1}{V}$ is upper bounded by the minimum binary length of a feasible solution $x^* > \mathbf{0}$ in $[0,1]^n$, if exists.

Lemma 2.2 *The basic procedure performs at most $O(n^3)$ iterations. An iteration of the basic procedure runs in $O(n)$ time.*

Proof. Since $\max y^{(s)} \geq \frac{1}{n}$, it follows that the stopping condition that implies (i) is reached

no later than $\|x^{(s)}\| \leq \frac{1}{2n\sqrt{n}}$. As $\frac{1}{\|x^{(s)}\|^2}$ increases by at least 1 in each iteration, the basic procedure requires at most $O(n^3)$ iterations.

At every iteration s the vector $y^{(s+1)}P_A$ is computed in $O(n)$ time as a convex combination of the already constructed vector $y^{(s)}P_A$ and the row p^T. It follows that an iteration of the loop runs in $O(n)$ time. ∎

If y^{out} is non-zero, then, since the squares of the potentials of the points $x^{(s)}$ increase by at least 1 in each iteration, the number of iterations is bounded by

$$\frac{1}{\|y^{out}P_A\|^2} - \frac{1}{\|y^{in}P_A\|^2} + 1.$$

Assume that the basic procedure results in (i) and returns a non-zero vector y^{out}. Consider the system with the coefficient matrix $A'' = AD$ where D is a diagonal matrix whose entries on the main diagonal are *non-positive* powers of 2. Let $y'' = \frac{y^{out}D}{y^{out}D\mathbf{1}}$. Let N be the number of the diagonal entries D_{ii} that are less than one, i.e., $N = |\{i|D_{ii} < 1\}|$. The lemma below will be needed for the analysis of the main algorithm that, under certain conditions, chooses $y^{in} = y''$ for the next call to the basic procedure.

Lemma 2.3

$$\frac{1}{\|y^{out}P_A\|^2} - \frac{1}{\|y''P_{A''}\|^2} \leq 8Nn^2.$$

Proof. Denote $y = y^{out}$ and $z = yA^T(AA^T)^{-1}$. We have

$$(zA'' - yD)P_{A''} = -yDP_{A''}.$$

Then, taking into account that the multiplication of a vector by $P_{A''}$ does not increase the length of the vector, we obtain

$$\|yDP_{A''}\| \leq \|zA'' - yD\| = \|(zA - y)D\| \leq \|zA - y\| = \|yP_A\|.$$

Since $y\mathbf{1} = 1$, we have

$$1 - \sum_{i:D_{ii}<1} y_i \leq yD\mathbf{1}.$$

It follows that

$$\left(1 - \sum_{i:D_{ii}<1} y_i\right) \cdot \frac{1}{\|yP_A\|} \leq \frac{yD\mathbf{1}}{\|yDP_{A''}\|} = \frac{1}{\|y''P_{A''}\|}.$$

Observe that, because y satisfies the condition of the while-loop,

$$y_i < \max y < 2 \cdot [yP_A]^+ \mathbf{1} \leq 2 \cdot \|yP_A\| \sqrt{n}$$

for all i. Then we can write

$$\frac{1}{\|yP_A\|^2} - \frac{1}{\|y''P_{A''}\|^2} \leq \left(2 \sum_{i:D_{ii}<1} y_i - \left(\sum_{i:D_{ii}<1} y_i \right)^2 \right) \frac{1}{\|yP_A\|^2} \leq \frac{4N\sqrt{n}}{\|yP_A\|} \leq 8Nn^2.$$

(We take into account that $\frac{1}{\|yP_A\|}$ is bounded by $2n\sqrt{n}$ due to the fact that the condition of the loop would not hold otherwise.) ∎

2.3 Main algorithm

Now we formulate the main algorithm. It repeats the calls to the basic procedure until either a solution is found or it can be proved that no positive solutions exist. The algorithm uses the bound $L_{\min}$ on the number of iterations that we have defined earlier.

Algorithm 2.1 LP ALGORITHM

Input: A system $Ax = \mathbf{0}, x \geq \mathbf{0}$.

Output: Either a solution $x > \mathbf{0}$ of the system or a proof that there are no such solutions.

$t := 0$;

$y^{in} := \mathbf{1}/n$;

$y' = y^{in}$; (The y' will store y^{out} at the most recent iteration with $y^{out} \neq \mathbf{0}$.)

$M^{(t)} = I$; (Matrix $M^{(t)}$ reflects the divisions of columns.)

$D = I$;

(Matrix D reflects the divisions beginning with the most recent iteration with $y^{out} \neq \mathbf{0}$.)

$A^{(0)} := A$;

repeat

Call the BASIC PROCEDURE for $A^{(t)}$ and y^{in};

if the call results in (iii) **then** return $x = M^{(t)}x^*$;

if the call results in (ii) **then** return (ii);

(The BASIC PROCEDURE results in (i).)

Construct $A^{(t+1)}$ by dividing the kth column of $A^{(t)}$ by 2;

Construct $M^{(t+1)}$ by dividing the kth column of $M^{(t)}$ by 2;

if $y^{out} \neq \mathbf{0}$ **then**

$y' := y^{out}$;

$D = I$;

end

Divide the kth column of D by 2;

$y^{in} := \frac{y'D}{y'D\mathbf{1}}$;

$t := t + 1$;

until $t > L_{\min}$

Decide that there are no solutions x with $x > \mathbf{0}$.

Remark. At the beginning of iteration t, $A^{(t)} = AM^{(t)}$. This means that $M^{(t)}x^*$ is a positive solution of $Ax = \mathbf{0}$ if x^* is a positive solution of $A^{(t)}x = \mathbf{0}$.

We call iteration t of the above algorithm fast if the basic procedure returns (i) with $y^{out} = \mathbf{0}$. If the iteration returns (i) and $y^{out} \neq \mathbf{0}$, it is called slow. Note that under this definition it can happen that the last iteration is neither fast nor slow.

Let t be some iteration and t' be a slow iteration, performed earlier than t, such that all iterations between t' and t (if there are any) are fast. Let D be the current matrix D at the beginning of iteration t. Then

$$M^{(t)} = M^{(t')}D$$

and

$$A^{(t)} = A^{(t')}D.$$

In other words, matrix D reflects the divisions of columns beginning with the most recent slow iteration.

Let T be the number of fast and slow iterations and F be the number of fast iterations. We have $T \leq L_{\min} + 1$.

Consider a slow iteration t. Let y^{in} be the vector y^{in} used for the input to the basic procedure. Let y^{out} be returned by the basic procedure. Denote

$$r'_t = \frac{1}{\|y^{in} P_{A^{(t)}}\|}$$

and

$$r''_t = \frac{1}{\|y^{out} P_{A^{(t)}}\|}.$$

The number of iterations of the basic procedure is bounded by

$$(r''_t)^2 - (r'_t)^2 + 1.$$

The total number iterations performed by the basic procedure in the course of the LP algorithm is bounded by

$$F + \sum_{t \text{ is slow}} ((r''_t)^2 - (r'_t)^2 + 1) + 1.$$

Let π enumerate all slow iterations in the order in which they appear in the course of the LP algorithm, i.e., the sequence $\pi(1), \ldots, \pi(T-F)$ represents the slow iterations. We obtain

$$\begin{aligned} &\textstyle\sum_t \text{ is slow}((r''_t)^2 - (r'_t)^2 + 1) = \sum_{q=1}^{T-F}((r''_{\pi(q)})^2 - (r'_{\pi(q)})^2 + 1) \\ &= (r''_{\pi(T-F)})^2 - (r'_{\pi(1)})^2 + 1 + \textstyle\sum_{q=1}^{T-F-1}((r''_{\pi(q)})^2 - (r'_{\pi(q+1)})^2 + 1). \end{aligned} \tag{4}$$

Note that $A^{(\pi(q+1))} = A^{(\pi(q))} D$ where D is the current matrix D at the beginning of iteration $\pi(q+1)$. Denote the number of diagonal entries $D_{ii} < 1$ by N_q. We have

$$N_q \leq \pi(q+1) - \pi(q)$$

because each iteration of the LP algorithm can transform only one column of the current matrix $A^{(t)}$. Consider the vector y^{in} used for the input of the basic procedure at the beginning of iteration $\pi(q+1)$. This vector is obtained as $y^{in} = \frac{y^{out} D}{y^{out} D \mathbf{1}}$ from y^{out} returned by the basic procedure at iteration $\pi(q)$ of the LP algorithm. Here, the matrix D is the current matrix D at the beginning of iteration $\pi(q+1)$. As mentioned before, we have $A^{(\pi(q+1))} = A^{(\pi(q))} D$. To apply Lemma 2.3, we need to replace A by $A^{(\pi(q))}$, A'' by $A^{(\pi(q+1))}$, y'' by y^{in}, and N by N_q. It follows from Lemma 2.3 that

$$(r''_{\pi(q)})^2 - (r'_{\pi(q+1)})^2 \leq 8 N_q n^2.$$

Then, taking into account that the sum $\sum_{q=1}^{T-F-1} N_q$ is bounded by T, by Lemma 2.3 we obtain that

$$\sum_{q=1}^{T-F-1} ((r''_{\pi(q)})^2 - (r'_{\pi(q+1)})^2 + 1) \leq \sum_{q=1}^{T-F-1} (8N_q n^2 + 1) \leq 8n^2 T + T.$$

Then (4) implies

$$\sum_{t \text{ is slow}} ((r''_t)^2 - (r'_t)^2 + 1) \leq 4n^3 + 8n^2 T + T + 1$$

because $r''_{\pi(T-F)} \leq 2n\sqrt{n}$. It remains to take into account that the basic procedure performs at most $O(n^3)$ iterations at the last iteration of the LP algorithm. It follows that the total number of iterations performed by the basic procedure is bounded by $O(n^3 + n^2 L_{\min})$.

Each iteration of the basic procedure runs in $O(n)$ time by Lemma 2.2. The initial point $x^{(0)}$ in the basic procedure is computed in time $O(n^2)$. The computation of $P_{A^{(t)}}$ requires at most $O(n^3)$ time. We obtain the following theorem:

Theorem 2.1 *The LP algorithm runs in time* $O(n^4 + n^3 L_{\min})$.

3 Polynomial algorithm

In this section the basic procedure is modified so that the intermediate numbers are of polynomial size and the theoretical estimates of the running time remain the same as that of the original basic procedure. Then we discuss the application to linear optimization problems and general systems of linear inequalities.

3.1 Modified basic procedure

To make the algorithm polynomial, we must ensure, along with a polynomial running time, that the sizes of the numbers are bounded by a polynomial in the size of A. For this purpose we replace α by its approximation $\alpha^{\#}$. To additionally reduce the sizes, we periodically compute $y^{(s+1)}$ as $y^{(s+1)} = y^{\#}$, where $y^{\#}$ is an approximation of $y = \alpha y^{(s)} + (1 - \alpha)\mathbf{e}_i$. This

kind of rounding implies the complete calculation of $y^{\#}P_A$ in time $O(n^2)$. The basic procedure performs this operation only once in each n iterations, which preserves the theoretical estimate of the running time.

In the original basic procedure, $x^{(s+1)}$ is equal to

$$x^{\perp} = \alpha x^{(s)} + (1-\alpha)p.$$

If $x^{\perp}$ does not satisfy the stopping condition that implies (i), the modified basic procedure replaces $x^{\perp}$ by a point $x^{\#}$, with $\|x^{\#}\| > \frac{1}{4n\sqrt{n}}$, that is sufficiently close to $x^{\perp}$ so that to guarantee

$$\frac{1}{\|x^{\perp}\|^2} - \frac{1}{\|x^{\#}\|^2} \leq \frac{1}{2}.$$

The theoretical estimates of the running time remain the same because the squares of potentials increase by at least $\frac{1}{2}$.

Since $\|x^{\perp}\| > \frac{1}{2n\sqrt{n}}$ (due to the fact that the current iteration is not the last one) and $\|x^{\#}\| > \frac{1}{4n\sqrt{n}}$, we can write

$$\frac{1}{\|x^{\perp}\|^2} - \frac{1}{\|x^{\#}\|^2} = \left(\|x^{\#}\| - \|x^{\perp}\|\right)\left(\frac{\|x^{\perp}\| + \|x^{\#}\|}{\|x^{\perp}\|^2\|x^{\#}\|^2}\right) < 64n^5(\|x^{\#}\| - \|x^{\perp}\|). \tag{5}$$

Let us consider the case where the rounding is performed by replacing α by $\alpha^{\#}$. In this case the modified procedure sets

$$x^{\#} := \alpha^{\#} x^{(s)} + (1-\alpha^{\#})p$$

where

$$\alpha^{\#} = \frac{1}{256n^5}\lfloor 256n^5\alpha \rfloor.$$

This value belongs to $[0,1]$ and can be computed in time $O(\log n)$ by a binary search. The choice of $\alpha^{\#}$ ensures that $\|x^{\#}\| - \|x^{\perp}\|$ is bounded by $\frac{1}{128n^5}$, which by (5) implies the bound of $\frac{1}{2}$ on $\frac{1}{\|x^{\perp}\|^2} - \frac{1}{\|x^{\#}\|^2}$ (note that $\|x^{\#}\| > \frac{1}{4n\sqrt{n}}$ because $\|x^{\perp}\| > \frac{1}{2n\sqrt{n}}$).

Now we consider the case in which the rounding is performed by replacing y by $y^{\#}$. In this case $x^{\#} = y^{\#}P_A$ where

$$y^{\#} = \mu\left\lfloor \frac{y}{\mu} \right\rfloor + \frac{1 - \mu\left\lfloor \frac{y}{\mu} \right\rfloor \mathbf{1}}{n}\mathbf{1}.$$

Here, μ is the parameter defined below and $\lfloor \cdot \rfloor$ denotes the rounding of each component down to an integer. Notice that $y^{\#}\mathbf{1} = 1$ and $|y^{\#}_j - y_j| \le \mu$ for all j. Taking into account that the norms of the columns of P_A are not greater than 1, we conclude that

$$|\|x^{\#}\| - \|x^{\perp}\|| \le \|x^{\#} - x^{\perp}\| = \|(y^{\#} - y)P_A\| \le n\mu.$$

Then, to obtain the bound of $\frac{1}{2}$ on the right-hand side of (5), it suffices to set

$$\mu := \frac{1}{256n^6}.$$

Notice that, with this value of μ, the components of $y^{\#}$ are integral multiples of $\frac{\mu}{n}$ (because 1 is an integral multiple of μ). Each component of $y^{\#}$ can be calculated in time $O(\log n)$. Therefore $y^{\#}$ can be constructed in $O(n \log n)$ time.

The above rounding technique leads to the respective modification of the basic procedure:

Procedure 3.1 MODIFIED BASIC PROCEDURE

Input: A matrix A and a row vector $y^{in} \ge \mathbf{0}$ with $y^{in}\mathbf{1} = 1$.

Output: One of the objects (i), (ii), or (iii), and a row vector y^{out} in case (i).

$y^{(0)} := y^{in}$;

Construct P_A and $y^{(0)}P_A$;

$x^{(0)} := P_A(y^{(0)})^T$;

$s := 0$;

while $\max y^{(s)} < 2 \cdot [(x^{(s)})^T]^+\mathbf{1}$

  if $x^{(s)} > \mathbf{0}$ **then** return $x^{(s)}$;

  else find i with $x^{(s)}_i \le 0$;

  Let p be the ith column of P_A;

  if $\mathbf{0}$ belongs to $[x^{(s)}, p]$ **then** set $k := i$ and return (ii);

  Compute α;

  $y := \alpha y^{(s)} + (1 - \alpha)\mathbf{e}_i$;

  $x^{\perp} = \alpha x^{(s)} + (1 - \alpha)p$;

  if $\max y < 2 \cdot [(x^{\perp})^T]^+\mathbf{1}$

then

if n divides s **then**

$y^{(s+1)} := y^{\#}$;

$x^{(s+1)} := P_A(y^{\#})^T$;

else

$y^{(s+1)} := \alpha^{\#} y^{(s)} + (1 - \alpha^{\#})\mathbf{e}_i$;

$x^{(s+1)} := \alpha^{\#} x^{(s)} + (1 - \alpha^{\#})p$;

end

else

$y^{(s+1)} := y$;

$x^{(s+1)} := x^{\perp}$;

end

$s := s + 1$;

end

Find k with $y_k^{(s)} = \max y^{(s)}$;

if $s \geq 1$ **then** return (i) and $y^{out} = y^{(s-1)}$ **else** return (i) and $y^{out} = \mathbf{0}$.

Remark. The vector $y^{\#}$ is computed in $O(n \log n)$ time. This is needed only when n divides s. (Note that n divides 0.) In this case the computation of $y^{(s+1)}P_A$ requires $O(n^2)$ time, which however does not influence the theoretical estimate of the running time. In $O(1)$ time it can be checked if n divides s. For this we can use an additional counter which is set to 0 after every n iterations.

The calculation of $\alpha^{\#}$ can be performed in $O(\log n)$ time. Note that not only $\alpha^{\#}$ but also $1 - \alpha^{\#}$ is a multiple of μ. Both $\alpha^{\#}$ and $1 - \alpha^{\#}$ can be represented in the form $d\mu$ where d is an integer such that $0 \leq d \leq 256n^6$.

The theoretical estimate of the number of iterations remains the same because the squares of the potentials increase by at least $\frac{1}{2}$. Note that Lemma 2.3 remains unchanged with respect to the modified procedure. So the LP algorithm with the modified basic procedure runs in time $O(n^4 + n^3 L_{\min})$.

At iteration s, if it is not the last one and $s \geq 1$, the components $y_j^{(s)}$, $j = 1, \ldots, n$, have the form

$$y_j^{(s)} = \sum_{q=1}^{s-n\lfloor s/n \rfloor+1} d_{jq} \left(\frac{\mu}{n}\right)^q.$$

Here, d_{jq} are nonnegative integers such that $d_{jq} \leq 256^q n^{7q}$. We can write

$$y_j^{(s)} = \left(\frac{\mu}{n}\right)^n \sum_q d_{jq} \left(\frac{n}{\mu}\right)^{n-q}.$$

It follows that the sizes of the components of $y^{(s)}$ with $s \geq 1$, with the exception of maybe the last iteration, are bounded by $O(n \log n)$ because the upper bound for q in the above sum does not exceed n and therefore $\left(\frac{n}{\mu}\right)^{n-q}$ are integers. Then, because the sizes of the entries of P_A are polynomially bounded in the size of A, the sizes of all numbers in the course of the basic procedure are polynomially bounded in the size of A and in the size of the input vector y^{in}.

Taking into account that matrix A is rescaled in the course of the LP algorithm, we formulate the following theorem:

Theorem 3.1 *The LP algorithm, if it uses the modified basic procedure, is a polynomial algorithm whose running time is bounded by* $O(n^4 + n^3 L_{\min})$.

Proof. The running time follows from the previous discussion. Let us give some more details concerning the sizes of the numbers. Consider iteration t of the LP algorithm. The size of y^{in} is polynomially bounded in $L_{\min}$ and in n. The sizes of the entries of $P_{A^{(t)}}$ are polynomially bounded in size$(A^{(t)})$. The size of $A^{(t)}$ is bounded by $O(\text{size}(A)+mL_{\min})$ because $t \leq L_{\min}+1$. The value $L_{\min}$ is bounded by a polynomial in the size of A. It follows that the sizes of all the numbers in the course of the algorithm are polynomially bounded. ∎

Corollary 3.1 *Let* $Ax = b, x \geq \mathbf{0}$, *have a solution* $x > \mathbf{0}$ *in the cube* $[0, \rho]^n$, *where* $\rho > 0$ *is an integer. This system can be solved in time*

$$O(n^3(S + n \cdot size(\rho)))$$

where S *is the minimum size of a solution* $x > \mathbf{0}$ *in* $[0, \rho]^n$.

Proof. Consider the equivalent system $Ax = \frac{b}{\rho}, x \geq \mathbf{0}$. The homogenization of the system leads to a system having a positive solution, in the unit cube, whose size is bounded by $O(S + n \cdot \text{size}(\rho))$. The size of any positive solution in the unit cube gives an upper bound on the number of iterations of the LP algorithm. ∎

If $L_{\min}$ is bounded by $O(\text{size}(A))$, we obtain at least the running time of $O(n^3 \cdot \text{size}(A))$ which resembles the theoretical running time (although not fully reflecting the practical running time that can be much better) of interior-point methods such as Renegar's algorithm [10] and Karmarkar's algorithm [7]. A system of linear inequalities may have the property that if there exists a feasible solution in the unit cube then for any i there is one with $x_i \geq \varepsilon$ where $\varepsilon < 1$ is a constant. In this case we can choose $L_{\min}$ bounded by $O(n)$, which gives the running time $O(n^4)$. On the other hand, $O(n \cdot \text{size}(A))$ is a trivial upper bound for $L_{\min}$. If we used it, we would get the estimate of the running time of the ellipsoid method (see the original paper of Khachiyan [8] and the review by Todd [12]).

3.2 General linear systems and linear optimization problems

Every system of linear inequalities can be represented in the form

$$Ax = b, x \geq \mathbf{0}.$$

If there exist variables that are equal to zero in all feasible solutions, then the basic procedure, applied to the respective homogeneous system, will always result in (i) or (ii). This leads to the following theorem.

Theorem 3.2 *Let L denote the binary size of the above system. A solution in the relative interior of the feasible set (i.e., a solution with the maximum number of positive components), as well as a linear system defining the affine hull of the feasible set, can be found in time $O(n^4 L)$.*

Proof. Consider the respective homogeneous system

$$Ax = b\xi, x \geq \mathbf{0}, \xi \geq 0.$$

Using Cramer's theorem we can choose a suitable value K bounded by $O(L)$ such that 2^{-K} would represent a lower bound on positive components of vertices of the polytope being the intersection of the cone of feasible solutions of the homogeneous system with the unit cube. If the number of divisions of the same column in the course of the LP algorithm exceeds K, then the respective variable is zero in all feasible solutions of the system. If this variable is ξ, the original system is infeasible. Otherwise, we eliminate the column from the current system and apply the LP algorithm to the obtained system. We repeat the calls to the LP algorithm until a positive solution $(x^*; \xi^*)$ of the current system is found or it is proved that no non-trivial solutions exist. In the former case, taking $\frac{x^*}{\xi^*}$ and setting the deleted variables to 0, we construct a solution in the relative interior of the feasible set of the original system. The overall number of calls to the basic procedure is bounded by $O(Kn)$. This implies the bound $O(n^4L)$ on the total number of arithmetic operations performed. Since the basic procedure will necessarily result in (i) or (ii) if there exists a variable being equal to zero in all feasible solutions of the current system, all the variables equal to zero in each solution of the original system will be identified. The system $Ax = b, x_i = 0$, where i runs over all the variables proved to be equal to zero, is a polyhedral description of the affine hull. ∎

It follows that we can use the LP algorithm to find a polyhedral description of the set of optimal solutions of an optimization problem

$$\begin{aligned} \max \quad & cx \\ \text{s.t.} \quad & Ax = b, \\ & x \geq \mathbf{0}. \end{aligned}$$

We can use the duality theorem to represent the optimization problem as a system $\bar{A}\bar{x} = \bar{b}, \bar{x} \geq \mathbf{0}$, whose solutions are primal-dual pairs of optimal solutions. Then we find a linear system defining the affine hull of the set of optimal solutions and add the remaining non-negativity constraints. This leads to

Corollary 3.2 *Let L denote the binary size of the above optimization problem. An optimal solution with the maximum number of positive components, as well as a polyhedral description of the set of optimal solutions, or a proof that no optimal solution exists can be found in*

time $O(n^4L)$.

3.3 Linear systems having a binary solution

Consider a system of the form

$$Ax = b, \mathbf{0} \leq x \leq \mathbf{1}.$$

Now we will develop a strongly polynomial algorithm to solve the following task: Either find a feasible solution of the system or prove that there is no binary solution, i.e., that there is no solution in $\{0,1\}^n$. Below we give a variant of the LP algorithm that performs this task in strongly polynomial time $O(n^4)$. The analysis of the running time is in many respects similar to that of Section 2.3. Therefore, to avoid unnecessary repetitions of formulas, we first use the original basic procedure. Then we replace it by the modified basic procedure to guarantee polynomial sizes of the numbers.

Let us represent the above system in a form which would be acceptable to the LP algorithm:

$$\begin{aligned} &Ax^1 - b\xi = \mathbf{0}, \\ &x^1 + x^2 - \mathbf{1}\xi = \mathbf{0}, \\ &x^{1,2} \geq \mathbf{0}, \\ &\xi \geq 0. \end{aligned}$$

Any solution with $\xi > 0$ can be translated into a solution of the original system and vice versa. The above system has the property that it possesses a nonzero binary solution if and only if the original system has a binary solution. By renaming the variables we rewrite the above system as $\bar{A}\bar{x} = \mathbf{0}, \bar{x} \geq \mathbf{0}$. Now we apply the basic procedure. If it is proved that $\bar{x}_k \leq 1/2$, then $\bar{x}_k$ must be zero for any binary solution. Then we can delete the kth column and apply the procedure again. Note that if k refers to ξ then the system $Ax = b, \mathbf{0} \leq x \leq \mathbf{1}$, has no binary solutions.

Assume that the basic procedure results in (i) and returns a non-zero vector y^{out}. Consider the system with the coefficient matrix $\bar{A}'' = \bar{A}D$ where D is obtained form the identity matrix by deleting some columns. Denote $y'' = \frac{y^{out}D}{y^{out}D\mathbf{1}}$. Let N be the number of deleted columns.

Let A'' be a full-rank matrix whose null space coincides with that of $\bar{A}''$. Let $\bar{n} = 2n + 1$ denote the number of columns of $\bar{A}$. The lemma below is analogous to Lemma 2.3.

Lemma 3.1

$$\frac{1}{\|y^{out}P_{\bar{A}}\|^2} - \frac{1}{\|y''P_{A''}\|^2} \leq 8N\bar{n}^2.$$

Proof. Denote $y = y^{out}$ and $z = y\bar{A}^T(\bar{A}\bar{A}^T)^{-1}$. We have

$$(z\bar{A}'' - yD)P_{A''} = -yDP_{A''}.$$

Then, taking into account that the multiplication of a vector by $P_{A''}$ does not increase the length of the vector, we obtain

$$\|yDP_{A''}\| \leq \|z\bar{A}'' - yD\| = \|(z\bar{A} - y)D\| \leq \|z\bar{A} - y\| = \|yP_{\bar{A}}\|.$$

Then we can write

$$(1 - (1 - yD\mathbf{1})) \cdot \frac{1}{\|yP_{\bar{A}}\|} \leq \frac{yD\mathbf{1}}{\|yDP_{A''}\|} = \frac{1}{\|y''P_{A''}\|}.$$

Observe that, because y satisfies the condition of the while-loop,

$$y_i < \max y < 2\|yP_{\bar{A}}\|\sqrt{\bar{n}}$$

for all i. Then

$$1 - yD\mathbf{1} = y\mathbf{1} - yD\mathbf{1} \leq 2N\|yP_{\bar{A}}\|\sqrt{\bar{n}}.$$

It follows that

$$\frac{1}{\|yP_{\bar{A}}\|^2} - \frac{1}{\|y''P_{A''}\|^2} \leq \left(2(1 - yD\mathbf{1}) - (1 - yD\mathbf{1})^2\right) \frac{1}{\|yP_{\bar{A}}\|^2} \leq \frac{4N\sqrt{\bar{n}}}{\|yP_{\bar{A}}\|} \leq 8N\bar{n}^2.$$

(We take into account that $\frac{1}{\|yP_{\bar{A}}\|}$ is bounded by $2\bar{n}\sqrt{\bar{n}}$.) ∎

Assume that the basic procedure results in (i) or (ii). Then $\bar{x}_k \leq 1/2$, which means that $\bar{x}_k$ must be zero for any binary solution. In this case we delete the respective column of $\bar{A}$. Let $\bar{A}'$ denote the obtained matrix. The deletion is performed in place of the division by 2 in the original LP algorithm. In contrast to a division, the deletion of a column can

reduce the rank of the matrix. Therefore, we delete an appropriate row of $\bar{A}'$ to obtain a full-rank matrix defining the same null space. Denote this matrix by A' and apply the basic procedure again. After some number of calls to the basic procedure we come to a full-rank matrix A'' whose null space coincides with that of the matrix $\bar{A}''$ obtained from $\bar{A}$ by deleting the respective columns. This leads to the following algorithm:

Algorithm 3.1 VARIANT OF THE LP ALGORITHM

Input: A system $Ax = b, \mathbf{0} \leq x \leq \mathbf{1}$.

Output: Either a solution x of the system or a decision that there are no binary solutions.

$t := 0$;

$y^{in} := \mathbf{1}/n$;

$y' = y^{in}$; (The y' will store y^{out} at the most recent iteration with $y^{out} \neq \mathbf{0}$.)

$M^{(t)} = I$; (The matrix reflects the deletions of columns.)

$D = I$; (Matrix D reflects deletions beginning with the latest iteration with $y^{out} \neq \mathbf{0}$.)

$\bar{A}^{(0)} := \bar{A}$;

repeat

Construct a full-rank matrix $\hat{A}^{(t)}$ defining the same subspace as $\bar{A}^{(t)}$;

Call the BASIC PROCEDURE for $\hat{A}^{(t)}$ and y^{in};

if the call results in (iii), **then**

Let $(\bar{x}^1; \bar{x}^2; \xi)$ be the solution obtained;

Return $x = \frac{M^{(t)}\bar{x}^1}{\xi}$;

end

(Otherwise, the BASIC PROCEDURE results in (i) or (ii).)

if k refers to ξ **then** decide that there are no binary solutions.

Construct $\bar{A}^{(t+1)}$ by deleting the kth column of $\bar{A}^{(t)}$;

Construct $M^{(t+1)}$ by deleting the kth column of $M^{(t)}$;

if $y^{out} \neq \mathbf{0}$ **then**

$y' := y^{out}$;

$D = I$;

end

Delete the kth column of D;

$y^{in} := \frac{y'D}{y'D\mathbf{1}}$;

$t := t + 1$;

until $t \geq n$

Decide that there are no binary solutions.

Remark. At the beginning of iteration t, $\bar{A}^{(t)} = \bar{A}M^{(t)}$. This means that $M^{(t)}x^*$ is a nonnegative solution of $\bar{A}\bar{x} = \mathbf{0}$ if x^* is a positive solution of $\bar{A}^{(t)}x = \mathbf{0}$. The multiplication of $M^{(t)}$ by x^* means obtaining an $\bar{n}$-vector from x^* by setting the removed variables to 0. If $t = n$, then there is i for which it has been proved that $x_i^1 = 0$ and $x_i^2 = 0$ in all binary solutions. But $x_i^1 = 0$ implies that $x_i^2 = 1$ in all nonzero binary solutions. This implies that the original system has no binary solutions. Therefore, it is sufficient to perform no more than n iterations.

Thus, we obtain a variant of the LP algorithm which either finds a solution or proves that the original system has no binary solutions. To analyze the running time, we define slow and fast iterations in exactly the same way as for the original LP algorithm. Now the matrices $M^{(t)}$ and D are as defined in the above variant of the LP algorithm.

Consider a slow iteration t. Let y^{in} be the vector y^{in} used for the input to the basic procedure. Let y^{out} be returned by the basic procedure. Denote

$$r'_t = \frac{1}{\|y^{in}P_{\hat{A}^{(t)}}\|}$$

and

$$r''_t = \frac{1}{\|y^{out}P_{\hat{A}^{(t)}}\|}.$$

Note that we have the same equation (4) as in Section 2.3.

Denote $N_q = \pi(q+1) - \pi(q)$. This is the difference between the number of columns of $\bar{A}^{(\pi(q+1))}$ and the number of columns of $\bar{A}^{(\pi(q))}$. Consider the vector y^{in} used for the input of the basic procedure at the beginning of iteration $\pi(q+1)$. This vector is obtained as $y^{in} = \frac{y^{out}D}{y^{out}D\mathbf{1}}$ from y^{out} returned by the basic procedure at iteration $\pi(q)$ of the variant of

the LP algorithm. Here, the matrix D is the current matrix D at the beginning of iteration $\pi(q+1)$. To apply Lemma 3.1, we need to replace $\bar{A}$ by $\hat{A}^{(\pi(q))}$, A'' by $\hat{A}^{(\pi(q+1))}$, y'' by y^{in} at the beginning of iteration $\pi(q+1)$, N by N_q, and $\bar{A}''$ by $\hat{A}^{(\pi(q))}D$. Note that the matrix $\hat{A}^{(\pi(q+1))}$ defines the same subspace as $\hat{A}^{(\pi(q))}D$. It follows from Lemma 3.1 that

$$(r''_{\pi(q)})^2 - (r'_{\pi(q+1)})^2 \leq 8N_q\bar{n}^2.$$

Taking into account that the sum $\sum_{q=1}^{T-F-1} N_q$ is bounded by n and applying Lemma 3.1 by analogy with Lemma 2.3 in Section 2.3, we obtain that the total number of iterations performed by the basic procedure is bounded by $O(n^3)$. Each iteration of the basic procedure runs in $O(n)$ time by Lemma 2.2. The initial point $x^{(0)}$ in the basic procedure is computed in time $O(n^2)$. The computation of the projection matrix runs in $O(n^3)$ time. So we prove

Theorem 3.3 *The variant of the LP algorithm either finds a solution to a system $Ax = b$, $\mathbf{0} \leq x \leq \mathbf{1}$, or proves that the system has no binary solutions in $O(n^4)$ time.*

Using the modified basic procedure in place of the original one, we preserve the same estimates of the running time and guarantee that the sizes of the numbers are polynomially bounded. That is, the variant of the LP algorithm that uses the modified basic procedure is a strongly polynomial algorithm that runs in time $O(n^4)$. Note that the time $O(n^4)$ improves the running time of the algorithm presented in [2] by a factor of n^2.

4 Remarks on implementation

In this section we discuss an implementation of our algorithm. The implemented version differs from the previously presented version in several important details. In particular, the vector p is now computed so that the behavior of the algorithm does not depend on the order of columns of the coefficient matrix.

We apply our algorithm to a system

$$Ax = b, x \geq \mathbf{0},$$

where b is a nonzero column vector. By introducing a new variable ξ we represent the system as

$$Ax - b\xi = \mathbf{0},$$
$$x \geq \mathbf{0},$$
$$\xi \geq 0.$$

Any positive solution with $\xi > 0$ can be translated back into a solution of the original system and vice versa. Further we denote the coefficient matrix $(A| - b)$ by $\bar{A}$.

4.1 Computational errors in projection matrices

The errors in the computation of projection matrices can be large if the square matrix needed to compute the projections is ill-conditioned. In this case a solution returned by the algorithm can be too far from the feasible set. A detailed investigation of related questions is out of the scope of this paper. We will only discuss the conditions that guarantee the correctness of the conclusions leading to (i).

Let the entries of a symmetric matrix P approximate the entries of $P_{\bar{A}}$ with an absolute error $\varepsilon > 0$. Consider the matrix $P^{\#} = P + \varepsilon \mathbf{1}\mathbf{1}^T$. This matrix is symmetric and has the property that $P^{\#} \geq P_{\bar{A}}$. Let the projection matrix $P_{\bar{A}}$ in the basic procedure be replaced by $P^{\#}$. Note that in place of finding $x_i^{(s)} \leq 0$ we now should find a violated inequality of the system $P^{\#}x > \mathbf{0}$. The condition

$$\max y^{(s)} \geq 2 \cdot [y^{(s)} P^{\#}]^{+} \mathbf{1}$$

implies

$$\max y^{(s)} \geq 2 \cdot [y^{(s)} P_{\bar{A}}]^{+} \mathbf{1},$$

which in turn implies (i). Since the norms of the rows and columns of $P^{\#}$ are bounded by $1 + 2\varepsilon\sqrt{n}$, it is sufficient that ε is not greater than $\frac{\sqrt{2}-1}{2\sqrt{n}}$ to guarantee that the norm of the vector p in the course of the basic procedure does not exceed $\sqrt{2}$. This means that the squares of potentials will increase by at least $\frac{1}{2}$ in each iteration.

4.2 Computational experiment

Our computational experiment is a preliminary test of the behavior of the basic procedure in the course of the LP algorithm of Section 2.3. We do not introduce any corrections related to the control over computational errors and the sizes of the numbers.

The projection matrices needed in the LP algorithm, except for the first iteration, are computed by the update formulas which follow from Sherman-Morrison-Woodbury formulas [4]. This update requires only $O(n^2)$ arithmetic operations. Of course, the computation by the update formulas can lead to a fast accumulation of computational errors. However, the algorithm produces solutions of acceptable quality for the random set of instances in our experiment.

For the implementation we use MATLAB version R2011b running on a standard computer.

To ensure that the initial coefficient matrix is of full rank, we use an orthogonalization function of MATLAB that constructs an equivalent full-rank system of equations.

During the preparation of this paper, we have learned from Professor Cornelis Roos, of the Delft University of Technology, that it can be more profitable to select p in the basic procedure as the following convex combination of all those columns $P^j_{\bar{A}}$ of $P_{\bar{A}}$ which correspond to non-positive values of $x^{(s)}$:

$$p = \frac{\sum_{j:x_j^{(s)} \leq 0} P^j_{\bar{A}}}{|\{j | x_j^{(s)} \leq 0\}|}$$

($y^{(s+1)}$ is computed respectively). This replacement preserves all the estimates except for the running time of an iteration. In the implementation we use exactly this rule. Although the running time of an iteration is not linear with this rule, the performance of the basic procedure on our set of instances seems to be even better compared to the rules which select a single column (below we present only the results of the experiment with the new rule). It is important to note that with this rule the behavior of the algorithm is independent of the order of columns.

In our experiment we consider yes-instances that were generated using the standard

random generator in MATLAB. The set consists of 110 instances divided in 11 subsets, each containing 10 instances. The subsets are grouped in five classes. In the first three classes the vectors b have the form $b = A(1, 2, \ldots, n)^T$, $b = A(1, 1/2, \ldots, 1/n)^T$, and $b = A(1, 1/2^2, \ldots, 1/n^2)^T$, respectively. That is, $(1, 2, \ldots, n)^T$, $(1, 1/2, \ldots, 1/n)^T$, and $(1, 1/2^2, \ldots, 1/n^2)^T$ are positive feasible solutions for the corresponding instances. In the fourth class, $b = Az$ where z is a 0-1 vector whose components are drawn from the uniform distribution on $\{0, 1\}$. Although the LP algorithm is actually designed to find positive solutions, it finds solutions of reasonable quality also in this case. In the fifth class, $b = Az$ where z is a 0-1 vector whose first $\lfloor\sqrt{n}\rfloor$ components are ones and the other components are zeros. The matrices consist of integers drawn from the uniform distribution on $\{-100, \ldots, 100\}$. The fourth class and each of the first two classes of instances consists of subsets of instances with 500, 1000, and 1500 variables. The third class, as well as the fifth class, consists of instances with 500 variables. The number of equations in each instance is the half of the number of variables, e.g., $m = 750$ for $n = 1500$. The number of instances in each subset is equal to 10.

The experimental results are summarized in Tables 1 and 2. Their columns correspond to the subsets of instances. In each table, the second line indicates the number of variables. The other numerical entries are approximations of the respective values obtained. In both the tables the third and the fourth lines represent the average and the maximum number of iterations, respectively, needed by the basic procedure in the course of the LP algorithm. The next two lines are the average and maximum numbers of iterations needed by the LP algorithm. The average time, in seconds, is the average time per instance required by the LP algorithm. The accuracy of the obtained solutions for each subset of instances is computed as the maximum absolute value of the components of the differences $Ax^* - b$ over the respective solutions x^*. The respective approximate values are represented in the form of powers 10^{-d}.

The standard LP solver of MATLAB is used in the default setting. (The coefficients in the objective function are set to 0.) For all classes of instances, the MATLAB demonstrates a stable performance with a single exception for the subset of instances of the first class with 1500 variables where MATLAB was not able to solve one instance (the respective entry

	$(1, 2, \ldots, n)$			$(1, 1/2, \ldots, 1/n)$			$(1/j^2)$
n	500	1000	1500	500	1000	1500	500
BP: Average	4.1	4.4	4.4	140	298	418	33.9
BP: Max	6	7	6	190	472	783	9957
LP: Average	1	1	1	1	1	1	293
LP: Max	1	1	1	1	1	1	1027
LP: Average time	**0.20**	**1.13**	**3.42**	1.80	23.60	97.10	219.30
LP: Accuracy	10^{-6}	10^{-5}	10^{-4}	10^{-10}	10^{-10}	10^{-10}	10^{-8}
ML: Average time	0.42	3.70	60.20(?)	**0.42**	**3.76**	**66.80**	**0.50**
ML: Accuracy	10^{-7}	10^{-6}	10^{-3}	10^{-6}	10^{-6}	10^{-7}	10^{-5}

Table 1: The first three classes of instances.

	Random 0-1 solution			$\sqrt{n}$ of ones
n	500	1000	1500	500
BP: Average	19.7	27.8	27.3	3840
BP: Max	25	36	33	23700
LP: Average	1	1	1	1.1
LP: Max	1	1	1	2
LP: Average time	**0.32**	**2.01**	**6.50**	54.50
LP: Accuracy	10^{-10}	10^{-9}	10^{-9}	10^{-8}
ML: Average time	0.40	3.77	63.00	**0.51**
ML: Accuracy	10^{-6}	10^{-6}	10^{-4}	10^{-6}

Table 2: The fourth and fifth classes

is marked with '?' in the table). The behavior of the LP algorithm seems to be rather sensitive to the structure of solutions. A somewhat better behavior is observed for the first class (Table 1) and for the fourth class (Table 2). Each instance of the first, second, and fourth class was solved in one iteration of the LP algorithm. Some instances in the third and the fifth classes were extremely hard for the LP algorithm, while MATLAB did not experience any problems with these instances. The accuracy of the constructed solutions is however comparable with that achieved by the LP solver of MATLAB.

Acknowledgments

I wish to thank two anonymous referees and Cornelis Roos (the Delft University of Technology) for their detailed comments and for suggesting several important improvements. This work is supported by the DFG (Deutsche Forschungsgemeinschaft) research grant CH 1133/1-1.

References

[1] Agmon, Sh. 1954. The relaxation method for linear inequalities. *Canad. J. Math.* **6**, 382-392.

[2] Chubanov, S. 2012. A strongly polynomial algorithm for linear systems having a binary solution. *Mathematical Programming* **134**: 533-570.

[3] Edmonds, J. 1967. Systems of distinct representatives and linear algebra. *J. Res. Nat. Bur. Standards* **71 B**, 241-245.

[4] Hager W.W. 1989. Updating the inverse of a matrix. *SIAM Review* **31**, 221-239.

[5] Goffin, J.L. 1980. The relaxation method for solving systems of linear inequalities. *Mathematics of Operations Research* **5**, 388-414.

[6] Goffin, J.L. 1982. On the non-polynomiality of the relaxation method for systems of linear inequalities. *Mathematical Programming* **22**, 93-103.

[7] Karmarkar, N. 1984. A new polynomial-time algorithm for linear programming. *Combinatorica* **4**, 353-395.

[8] Khachiyan, L.G. 1979. A polynomial algorithm in linear programming. *Dokl. Akad. Nauk SSSR 244* (English translation: Soviet Math. Dokl. **20**, 191-194).

[9] Motzkin, Th., and Schoenberg, I. J. 1954. The relaxation method for linear inequalities. *Canad. J. Math.* **6** 393-404.

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

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

[12] Todd, M. 2002. The many facets of linear programming. *Mathematical Programming* **91**, 417-436.