# A strongly polynomial algorithm for linear systems having a binary solution

Sergei Chubanov

*Institute of Information Systems at the University of Siegen, Germany*

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

7th March 2011

**Abstract**

This paper proposes a strongly polynomial algorithm which either finds a solution of a linear system $Ax = b, \mathbf{0} \leq x \leq \mathbf{1}$, or correctly decides that the system has no 0,1-solutions. The algorithm can be used as the basis for the construction of a polynomial algorithm for linear programming.

## 1 Introduction

This paper describes a strongly polynomial algorithm which either finds a solution to a linear system $Ax = b, \mathbf{0} \leq x \leq \mathbf{1}$, where the coefficients are integers, or correctly decides that the system has no 0,1-solutions.

The algorithm can be used as the basis for the construction of a polynomial algorithm for linear programming (see [2] for details). This gives an algorithm which substantially differs from the well-known polynomial algorithms like, for instance, the ellipsoid method

by Khachiyan [5] or the projective algorithm by Karmarkar [4]. (The detailed survey of algorithms for linear programming can be found in Todd [8].)

The most important properties on which our method is based are the (Hahn-Banach) separation theorem for disjoint convex sets and the Cauchy-Schwarz inequality.

The core of our approach is a divide-and-conquer algorithm which may be considered as a generalization of the well-known relaxation method (see Agmon [1] and Motzkin and Shoenberg [6]). Unlike the classical relaxation method, the divide-and-conquer algorithm projects current solutions not only onto the half-spaces corresponding to original constraints, but also onto those which correspond to valid inequalities constructed in the course of the computation.

The main part of the paper consists of four sections. Section 2 discusses systems of inequalities and related questions. Section 3 explains the divide-and-conquer algorithm. Section 4 studies the effect of the parametrization of the right-hand side of the system. The obtained properties imply the strongly polynomial algorithm which is discussed in Section 5.

# 2 Systems of linear inequalities

We consider a system

$$Ax = b, Cx \leq d,$$

where $A$ is an $m \times n$ matrix, and $C$ is a $k \times n$ matrix. The entries of $A$, $b$, $C$, and $d$ are assumed to be integers. We also assume that the matrices have no zero rows. The $i$-th row of $A$ will be denoted by $a_i$ and the $i$-th row of $C$ will be denoted by $c_i$.

## 2.1 Induced inequalities

We say that an inequality $hx \leq \delta$ is induced by a system of linear equations and inequalities if and only if $hx \leq \delta$ is a linear combination of the equations plus a nonnegative linear

combination of the inequalities. That is, if $hx \leq \delta$ is induced by $Ax = b, Cx \leq d$, then

$$h = \sum_{l=1}^{m} \lambda_l a_l + \sum_{i=1}^{k} \alpha_i c_i$$

and

$$\delta = \sum_{l=1}^{m} \lambda_l b_l + \sum_{i=1}^{k} \alpha_i d_i$$

where $\lambda_l$ and $\alpha_i$ are some real coefficients and $\alpha_i \geq 0$ for all $i$. We denote the set of feasible solutions by

$$P = \{x | Ax = b, Cx \leq d\}.$$

If $hx \leq \delta$ is induced by the system, then $hx \leq \delta$ is a valid inequality, that is, it holds for every $x$ in $P$. The definition of induced inequalities makes sense even if $P$ is empty.

## 2.2 Task

Let $\| \cdot \|$ denote the Euclidean norm. The set

$$B(z, r) = \{x \in \mathbb{R}^n \mid \|x - z\| < r\}$$

is an open ball centered at $z \in \mathbb{R}^n$ with a radius $r$. Let $\mathbf{0}$ be an all-zero column vector and $\mathbf{1}$ be an all-one column vector. (The number of the components of both $\mathbf{0}$ and $\mathbf{1}$ is specified by the context.) The divide-and-conquer algorithm will be needed to solve the following task:

Given $(z, r, \varepsilon)$, either find an $\varepsilon$-approximate solution $x^*$ of the system in the sense that

(i) $Ax^* = b$, $Cx^* \leq d + \varepsilon\mathbf{1}$,

or a nonzero row vector $h$ and a value $\delta$ such that

(ii) $hx > \delta$ for all $x \in B(z, r)$ and the inequality $hx \leq \delta$ is induced by the system $Ax = b, Cx \leq d$.

The values $r$ and $\varepsilon$ are assumed to be positive. If it is clear from the context which linear system is considered, we will use only the tuple $(z, r, \varepsilon)$ to denote the corresponding task. We call $z$ the center and $r$ the radius because they define the ball $B(z, r)$ in the part (ii).

A solution to the task is either $x^*$ with (i) or $(h, \delta)$ with (ii). In other words, a solution is either an $\varepsilon$-approximate solution of the system or an induced inequality $hx \leq \delta$ such that $hx > \delta$ for all $x \in B(z, r)$. If the polyhedron $P$ is nonempty, then it is separated from $B(z, r)$ by the hyperplane $\{x|hx = \delta\}$.

**Remark.** *The condition (ii) is equivalent to the following one:*

*$hx \leq \delta$ is induced by the system $Ax = b, Cx \leq d$, and the distance from $z$ to the half-space $\{x|hx \leq \delta\}$ is not less than $r$. The latter condition holds if and only if*

$$\frac{hz - \delta}{\|h\|} \geq r.$$

*This inequality guarantees that $hz > \delta$, because $r > 0$, and that the distance between $z$ and the hyperplane $\{x|hx = \delta\}$ is not less than $r$. This ensures $hx > \delta$ for all $x$ in $B(z, r)$.*

## 2.3 Projections onto the affine subspace

We will assume that $A$ has rank $m$. Thus, the $m \times m$ matrix $AA^T$ is invertible. The orthogonal projection of $z \in \mathbb{R}^n$ onto the affine subspace $\{x|Ax = b\}$ is given by the mapping

$$p(z) = z + A^T(AA^T)^{-1}(b - Az).$$

The distance between $z$ and the affine subspace is equal to $\|p(z) - z\|$.

**Proposition 2.1** *If $\|p(z) - z\| \geq r$, then $B(z, r)$ is separated from the affine subspace $\{x|Ax = b\}$ by the hyperplane $\{x|hx = \delta\}$ where $h = (z - p(z))^T$ and $\delta = hp(z)$. The inequality $hx \leq \delta$ is induced by $Ax = b, Cx \leq d$. For all $x$ in $B(z, r)$, there holds $hx > \delta$.*

**Proof.** We have

$$hz - \delta = (z - p(z))^T(z - p(z)) = \|z - p(z)\|^2 \geq r^2 > 0.$$

It follows that $z$ violates the inequality $hz \leq \delta$. We have $\|h\| \geq r$ because $h = (z - p(z))^T$. Since $\delta = hp(z)$, the distance between $z$ and $\{x|hx = \delta\}$ is equal to

$$\frac{hz - \delta}{\|h\|} = \frac{h(z - p(z))}{\|h\|} = \frac{\|h\|^2}{\|h\|} = \|h\| \geq r.$$

Then, since the distance is not less than $r$ and $hz > \delta$, it follows that $hx > \delta$ for all $x$ in $B(z, r)$. Note that

$$h = (z - p(z))^T = -(A^T(AA^T)^{-1}(b - Az))^T$$

which means that $h$ is a linear combination of the rows of $A$. Denoting $a = (AA^T)^{-1}(b - Az)$, we get

$$h = -(A^T a)^T = -a^T A$$

and

$$\delta = hp(z) = -a^T Ap(z) = -a^T b.$$

Note that $a$ is a column vector. That is, $hx = \delta$ is a linear combination of the equations $Ax = b$. It follows that $hx \leq \delta$ is induced by $Ax = b, Cx \leq d$. ∎

## 2.4 Procedure for an elementary case

We now prove that if $r$ is sufficiently small, then the task $(z, r, \varepsilon)$ can be solved by evaluating the distances from the center to the affine subspace and the half-spaces. Let $c_{\max}$ be a row of $C$ of the maximum length.

**Proposition 2.2** *The following statements are true:*

*1) If* $r \leq \frac{\varepsilon}{2\|c_{\max}\|}$ *and* $\frac{c_i z - d_i}{\|c_i\|} < r$, *then* $c_i x < d_i + \varepsilon$ *for all* $x$ *in the ball* $B(z, r)$.

*2) If* $\frac{c_i z - d_i}{\|c_i\|} \geq r$, *then* $c_i x > d_i$ *for all* $x$ *in the ball* $B(z, r)$.

**Proof.** Assume that $x$ belongs to the ball. Then $x = z + v$ where $\|v\| < r$. By the Cauchy-Schwarz inequality, $|c_i v| \leq \|c_i\| \cdot \|v\| < \|c_i\| \cdot r$.

1) Since $r \leq \frac{\varepsilon}{2\|c_{\max}\|}$, we have $|c_i v| < \frac{\varepsilon\|c_i\|}{2\|c_{\max}\|}$. Then $c_i x - d_i = c_i z - d_i + c_i v < \|c_i\| \cdot r + c_i v \leq \frac{\varepsilon\|c_i\|}{2\|c_{\max}\|} + |c_i v| < \frac{\varepsilon\|c_i\|}{\|c_{\max}\|} \leq \varepsilon$.

2) $c_i x - d_i = c_i z - d_i + c_i v \geq \|c_i\| \cdot r + c_i v \geq \|c_i\| \cdot r - |c_i v| > \|c_i\| \cdot r - \|c_i\| \cdot r = 0$.

∎

Propositions 2.1 and 2.2 prove that if $r \leq \frac{\varepsilon}{2\|c_{\max}\|}$, then the following procedure produces a solution to the task $(z, r, \varepsilon)$ :

**Procedure 2.1** ELEMENTARY PROCEDURE

**if** $\|p(z) - z\| \geq r$ **then** return $h = (z - p(z))^T$ and $\delta = hp(z)$;

**else**

**if** $\frac{c_i z - d_i}{\|c_i\|} < r$ for all $i$ **then** return $x^* = p(z)$;

**else** return $h = c_i$ and $\delta = d_i$ such that $\frac{c_i z - d_i}{\|c_i\|} \geq r$.

**end if**

First, the procedure verifies whether $\|p(z) - z\| \geq r$. If it is so, then by Proposition 2.1 $h = (z - p(z))^T$ and $\delta = hp(z)$ deliver a solution to the task. If not, then $p(z)$ belongs to $B(z, r)$. In this case, if $\frac{c_i z - d_i}{\|c_i\|} < r$ for all $i = 1, \ldots, k$, the projection $p(z)$ gives an $\varepsilon$-approximate solution of $Ax = b, Cx \leq d$, by part 1 of Proposition 2.2, because $r \leq \frac{\varepsilon}{2\|c_{\max}\|}$ and $p(z) \in B(z, r)$. If $\frac{c_i z - d_i}{\|c_i\|} \geq r$ for some $i$, the procedure returns $(c_i, d_i)$ such that $\frac{c_i z - d_i}{\|c_i\|} \geq r$. By part 2 of Proposition 2.2, $c_i x > d_i$ for all $x$ in $B(z, r)$. It follows that $(h, \delta) = (c_i, d_i)$ satisfies (ii) because $c_i x \leq d_i$ is induced by the system. Thus the above procedure solves the task if $r$ is sufficiently small.

## 2.5 Separation of convex sets

The D&C ALGORITHM reduces the original task to a number of tasks falling under the elementary case. The reduction is based on the following argument. Let $h_1$ and $h_2$ be such that $h_1 \neq -\gamma h_2$ for every $\gamma > 0$. This is a sufficient condition for the set

$$Y = \{x \mid h_1 x \leq \delta_1,\ h_2 x \leq \delta_2\}$$

of all $x$ satisfying both $h_1 x \leq \delta_1$ and $h_2 x \leq \delta_2$ to be nonempty. The set $Y$ is convex. If $X \cap Y = \emptyset$ for a convex set $X$, then there is a hyperplane separating $X$ from $Y$. In other words, there is an inequality $hx \leq \delta$ which is satisfied by all points in $Y$ and violated by all points of $X$. Such inequality can be given by a convex combination of the inequalities $h_1 x \leq \delta_1$ and $h_2 x \leq \delta_2$. Thus, we have

**Proposition 2.3** *Let $X$ be a convex set in the Euclidean space. If every point $x$ in $X$ violates at least one of the inequalities $h_1 x \leq \delta_1$ or $h_2 x \leq \delta_2$, then there exists $\alpha \in [0, 1]$ such*

*that*

$$(\alpha h_1 + (1-\alpha)h_2)\, x > \alpha\delta_1 + (1-\alpha)\delta_2$$

*for all $x$ in $X$.*

**Proof.** The proposition is a direct implication of some well-known facts. Since $h_1 x > \delta_1$ or $h_2 x > \delta_2$ for every $x \in X$, it follows that $X$ and $Y$ have no common points. From the theory of convex sets, which we apply now to the Euclidean space, it follows that there exists a hyperplane separating $X$ from $Y$. That is, there is a row vector $h'$ and a value $\delta'$ such that $h'x > \delta'$ for all $x$ in $X$ and $h'x \leq \delta'$ for all $x$ in $Y$. The affine variant of Farkas' Lemma implies that there is $\delta'' \leq \delta'$ such that $h'x \leq \delta''$ is a nonnegative combination of the inequalities $h_1 x \leq \delta_1$ and $h_2 x \leq \delta_2$. That is, $h' = \gamma_1 h_1 + \gamma_2 h_2$ and $\delta'' = \gamma_1\delta_1 + \gamma_2\delta_2$ where $\gamma_1$ and $\gamma_2$ are nonnegative coefficients whose sum is nonzero. For all $x$ in $X$ we have $h'x > \delta' \geq \delta''$ and for all $x$ in $Y$ we have $h'x \leq \delta''$. Dividing both sides of the inequality $h'x \leq \delta''$ by $\gamma_1 + \gamma_2$, we get the inequality $(\alpha h_1 + (1-\alpha)h_2)x \leq \alpha\delta_1 + (1-\alpha)\delta_2$ where $\alpha \in [0,1]$. This inequality is violated by all $x$ in $X$. ∎

# 3 Divide-and-conquer algorithm

In this section we describe an algorithm which either solves the task or delivers two induced inequalities $h_1 x \leq \delta_1$ and $h_2 x \leq \delta_2$ such that $h_1 = -\gamma h_2$ for some positive value $\gamma$. In the latter case we say for short that the algorithm fails to find a solution or simply fails. The algorithm belongs to the class of divide-and-conquer algorithms because it is based on a multi-branched recursion.

At first we discuss a preliminary version which does not control the sizes of the numbers. Then we consider a modified version which rounds the numbers so that their sizes grow not too fast.

In general, the divide-and-conquer algorithm is exponential. It will, however, be used later on as a basic procedure in the polynomial algorithm. There, the divide-and-conquer algorithm is used for the tasks with $\varepsilon = 1$. If the constraints $Cx \leq d$ have the form $\mathbf{0} \leq x \leq \mathbf{1}$,

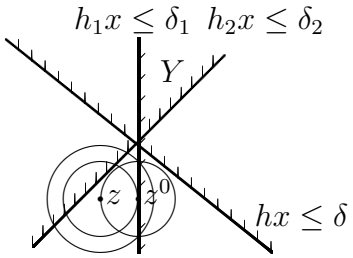


Figure 1: $Y = \{x \mid h_1x \le \delta_1, h_2x \le \delta_2\}$. $P$ is a subset of $Y$.

then the divide-and-conquer algorithm runs in polynomial time when being called from the main algorithm.

## 3.1 Preliminary divide-and-conquer algorithm

Consider a task $(z, r, \varepsilon)$. The algorithm works as follows. If the radius $r$ is sufficiently small, then the algorithm applies the ELEMENTARY PROCEDURE (Procedure 2.1). Otherwise, the algorithm specifies an appropriate smaller radius $r^\# < r$ and reduces the task $(z, r, \varepsilon)$ to tasks $(z, r^\#, \varepsilon)$ and $(z^0, r^\#, \varepsilon)$. This leads to the recursion. One step of the recursion is illustrated by Figure 1. At first the algorithm tries to solve $(z, r^\#, \varepsilon)$. If it finds an $\varepsilon$-approximate solution $x^*$ for $(z, r^\#, \varepsilon)$, then $x^*$ is a solution also for $(z, r, \varepsilon)$ because part (i) of the tasks refers to the same system. If the algorithm finds an induced inequality $h_1x \le \delta_1$ such that $h_1x > \delta_1$ for all $x$ in $B(z, r^\#)$, then it calculates the projection $z^0$ of $z$ onto the half-space $\{x|h_1x \le \delta_1\}$ and tries to solve $(z^0, r^\#, \varepsilon)$. If an $\varepsilon$-approximate solution $x^*$ is found, then $x^*$ is a solution also for the original task $(z, r, \varepsilon)$. If an induced inequality $h_2x \le \delta_2$ is found, such that $h_2x > \delta_2$ for all $x$ in $B(z^0, r^\#)$, then the algorithm constructs an induced inequality $hx \le \delta$ such that $hx > \delta$ for all $x$ in $B(z, r)$. For this purpose, the algorithm uses the fact that the appropriate choice of $r^\# \ge r/\sqrt{2}$ guarantees that

$$B(z, r) \cap \{x|h_1x \le \delta_1\} \subseteq B(z^0, r^\#). \tag{3.1}$$

This inclusion implies $h_2x > \delta_2$ for all those points $x$ in $B(z, r)$ which satisfy $h_1x \le \delta_1$. So every point in $B(z, r)$ violates at least one of the inequalities $h_1x \le \delta_1$ or $h_2x \le \delta_2$. Proposition 2.3, if applied to $X = B(z, r)$, claims that there is an inequality $hx \le \delta$ which

is a convex combination of the two inequalities and such that $hx > \delta$ for all $x$ in $B(z, r)$. As we will see later on, an appropriate convex combination can be calculated by explicit formulas. The inequality $hx \leq \delta$ is an induced inequality because it is a nonnegative linear combination of induced inequalities. So the algorithm solves the original task $(z, r, \varepsilon)$. Note that Proposition 2.3 is applicable if $h_1 \neq -\gamma h_2$ for all $\gamma > 0$. If this condition does not hold, we say that the algorithm fails to find a solution, or simply fails. The algorithm stops in this case.

Let $r^{\#}$ be equal to $\frac{1}{1+\theta}r$. Here, $\theta$ is a positive rational constant such that $1+\theta < \sqrt{2}$. We set $\theta = \frac{\theta'}{\theta''}$ where $\theta' = 2$ and $\theta'' = 5$. The inclusion (3.1) takes the form

$$B(z, r) \cap \{x | h_1 x \leq \delta_1\} \subseteq B\left(z^0, \frac{1}{1+\theta}r\right). \tag{3.2}$$

We prove the inclusion after the detailed description of the algorithm.

So the algorithm consists of four steps given below. The input is a task $(z, r, \varepsilon)$. The output is either an $\varepsilon$-approximate solution $x^*$ or an induced inequality $hx \leq \delta$ such that $hx > \delta$ for all $x \in B(z, r)$.

**Step 1.** If $r \leq \frac{\varepsilon}{2\|c_{\max}\|}$, then solve the task by the ELEMENTARY PROCEDURE (Procedure 2.1). (The solution is either an $\varepsilon$-approximate solution or an induced inequality.) Otherwise, go to step 2.

**Step 2.** Apply the algorithm recursively to the input $(z, \frac{1}{1+\theta}r, \varepsilon)$. If the recursive call results in an $\varepsilon$-approximate solution $x^*$, then return $x^*$, because $x^*$ is a solution also of the task $(z, r, \varepsilon)$. If the recursive call returns an induced inequality $h_1 x \leq \delta_1$, then go to step 3.

**Step 3.** Set

$$z^0 := z - \frac{h_1 z - \delta_1}{h_1 h_1^T} h_1^T.$$

This is the projection of $z$ onto the hyperplane $\{x | h_1 x = \delta_1\}$ (and at the same time onto the half-space $\{x | h_1 x \leq \delta_1\}$ because it is assumed that the recursive call at step 2 produces a correct output and $h_1 z > \delta_1$ therefore). Apply the algorithm recursively to $(z^0, \frac{1}{1+\theta}r, \varepsilon)$. If the recursive call results in an $\varepsilon$-approximate solution $x^*$, then return $x^*$. If the recursive call returns an induced inequality $h_2 x \leq \delta_2$, then go to step 4.

**Step 4.** At this step, it is assumed that the recursive calls at steps 2 and 3 have returned a correct output. That is, the induced inequalities $h_1x \leq \delta_1$ and $h_2x \leq \delta_2$ are such that $h_1x > \delta_1$ for all $x \in B(z, \frac{1}{1+\theta}r)$ and $h_2x > \delta_2$ for all $x \in B(z^0, \frac{1}{1+\theta}r)$. Under this assumption, we will prove that the algorithm either fails to find a solution or returns an induced inequality $hx \leq \delta$ such that $hx > \delta$ for all $x \in B(z, r)$. Taking into account $h_2x > \delta_2$ for all $x \in B(z^0, \frac{1}{1+\theta}r)$, we may see that the inclusion (3.2) guarantees $h_2x > \delta_2$ for all those points $x$ in $B(z, r)$ which satisfy $h_1x \leq \delta_1$. That is, each point $x$ in $B(z, r)$ violates at least one of the inequalities $h_1x \leq \delta_1$ or $h_2x \leq \delta_2$. If $h_1 \neq -\gamma h_2$ for all $\gamma > 0$, then the application of Proposition 2.3 to $X = B(z, r)$ proves that there is $\alpha$ in $[0, 1]$ such that

$$(\alpha h_1 + (1-\alpha)h_2)x > \alpha\delta_1 + (1-\alpha)\delta_2$$

for all $x \in B(z, r)$. Since the inequalities $h_1x \leq \delta_1$ and $h_2x \leq \delta_2$ are induced by the system, the inequality $hx \leq \delta$ with

$$h = \alpha h_1 + (1-\alpha)h_2$$

and

$$\delta = \alpha\delta_1 + (1-\alpha)\delta_2$$

is induced by the system as well. It follows that $(h, \delta)$ is a solution to the task $(z, r, \varepsilon)$. To find a value $\alpha \in [0, 1]$ giving $hx > \delta$ for all $x \in B(z, r)$, it is sufficient to solve the inequality

$$\frac{hz - \delta}{\|h\|} \geq r$$

with respect to $\alpha$. (See the remark in Subsection 2.2.) After the pseudocode describing the algorithm, it will be shown how to find $\alpha$ by a simple single-variable quadratic optimization.

**Proof of the inclusion (3.2).** Since $(h_1, \delta_1)$ is a solution of the task $(z, \frac{1}{1+\theta}r, \varepsilon)$, the intersection of $B(z, \frac{1}{1+\theta}r)$ with the half-space $\{x|h_1x \leq \delta_1\}$ is empty. It follows that the distance between $z$ and the hyperplane $\{x|h_1x = \delta_1\}$ is not less than $\frac{1}{1+\theta}r$. That is, taking into account that $h_1z - \delta_1 > 0$, we may write

$$\frac{h_1z - \delta_1}{\|h_1\|} \geq \frac{1}{1+\theta}r. \tag{3.3}$$

We now prove (3.2). Assume there exists a point $x \in B(z,r)$ with $h_1 x \le \delta_1$. The following chain of equations and inequalities proves that the point $x$ belongs to $B(z^0, \frac{1}{1+\theta}r)$ (here we use $\frac{1}{(1+\theta)^2} > \frac{1}{2}$, $h_1 x \le \delta_1$, and (3.3) ):

$$\begin{aligned}
\|x-z^0\|^2 &= (x-z^0)^T(x-z^0) = ((x-z)^T + \tfrac{h_1 z-\delta_1}{h_1 h_1^T} h_1)((x-z) + \tfrac{h_1 z-\delta_1}{h_1 h_1^T} h_1^T) \\
&= \|x-z\|^2 + 2\cdot \tfrac{h_1 z-\delta_1}{h_1 h_1^T}\cdot h_1(x-z) + \tfrac{(h_1 z-\delta_1)^2}{h_1 h_1^T} \\
&\le \|x-z\|^2 + 2\cdot \tfrac{h_1 z-\delta_1}{h_1 h_1^T}\cdot (\delta_1 - h_1 z) + \tfrac{(h_1 z-\delta_1)^2}{h_1 h_1^T} \\
&= \|x-z\|^2 - 2\cdot \tfrac{(h_1 z-\delta_1)^2}{h_1 h_1^T} + \tfrac{(h_1 z-\delta_1)^2}{h_1 h_1^T} \\
&= \|x-z\|^2 - \tfrac{(h_1 z-\delta_1)^2}{h_1 h_1^T} \\
&< r^2 - \tfrac{r^2}{(1+\theta)^2} < r^2 - \tfrac{r^2}{2} = \tfrac{r^2}{2} < \tfrac{r^2}{(1+\theta)^2}.
\end{aligned}$$

Thus, all points $x \in B(z,r)$ with $h_1 x \le \delta_1$ must be in the ball $B(z^0, \frac{1}{1+\theta}r)$. ∎

The pseudocode below provides a concise description of the algorithm.

**Algorithm 3.1** PRELIMINARY D&C ALGORITHM

**Input:** $(z, r, \varepsilon)$.

**Output:** Either $x^*$ with (i) or $(h, \delta)$ with (ii).

**1. if** $r \le \frac{\varepsilon}{2\|c_{\max}\|}$ **then**

Solve $(z, r, \varepsilon)$ by the ELEMENTARY PROCEDURE and return the solution.

**end if**

**2.** Run the PRELIMINARY D&C ALGORITHM with $\left(z, \frac{1}{1+\theta}r, \varepsilon\right)$.

**if** the recursive call returns $x^*$ with (i) **then** return $x^*$;

**else** let $(h_1, \delta_1)$ be returned by the recursive call.

**3.** $z^0 := z - \frac{h_1 z-\delta_1}{h_1 h_1^T} h_1^T$.

Run the PRELIMINARY D&C ALGORITHM with $\left(z^0, \frac{1}{1+\theta}r, \varepsilon\right)$.

**if** the recursive call returns $x^*$ with (i) **then** return $x^*$;

**else** let $(h_2, \delta_2)$ be returned by the recursive call.

**4. if** $h_1 = -\gamma h_2$ for some $\gamma > 0$ **then** STOP. The algorithm fails.

**else**

Find $\alpha$ such that $\frac{hz-\delta}{\|h\|} \ge r$ where $h = \alpha h_1 + (1-\alpha)h_2$ and $\delta = \alpha\delta_1 + (1-\alpha)\delta_2$.

Return $(h, \delta)$.

**end if**

Note that steps 2 and 3 are formulated correctly because if the algorithm is still being executed after a recursive call then the recursive call has returned a solution to the corresponding task.

Further, a repetition of the steps 1-4 is called an *iteration*. An iteration may be interrupted by other iterations because of the recursive structure of the algorithm. The first iteration is applied to the initial input $(z, r, \varepsilon)$. This iteration starts earlier than the others and finishes later. Note that the algorithm may fail at any iteration. Whenever it fails, the entire computation stops. As we will see in Section 4, the inequalities $h_1 x \leq \delta_1$ and $h_2 x \leq \delta_2$ obtained by the respective iteration contain useful information which makes it possible to decide that some of the inequalities can be replaced with equations. Such conclusion is based on the fact that $h_1 = -\gamma h_2$ in the case of failure.

**Remark.** *If we need to consider another iteration than the first one, we will denote the current input by $(\hat{z}, \hat{r}, \varepsilon)$ to distinguish between the initial input and the input to which the current iteration is applied.*

**Remark.** *Note that the algorithm does not add any inequality to the system in question. That is, every recursive call is applied to the same system of inequalities. An iteration produces at most one induced inequality. The information about previously constructed inequalities $h_1 x \leq \delta_1$ and $h_2 x \leq \delta_2$ is not necessary at the end of an iteration. So this information is deleted.*

**The implementation of step 4.** Denote $s_1 = h_1 z - \delta_1$ and $s_2 = h_2 z - \delta_2$. We have

$$hz - \delta = (s_1 - s_2)\alpha + s_2.$$

Thus the inequality

$$\frac{hz - \delta}{\|h\|} \geq r$$

is equivalent to

$$\frac{(s_1 - s_2)\alpha + s_2}{\|(h_1 - h_2)\alpha + h_2\|} \geq r. \tag{3.4}$$

The inequality guarantees that $(s_1 - s_2)\alpha + s_2$ is positive because $r$ is positive. The division is correct because there is no $\gamma > 0$ with $h_1 = -\gamma h_2$ which implies that $(h_1 - h_2)\alpha + h_2$ is a nonzero vector for every $\alpha \in [0, 1]$ because $h_1$ and $h_2$ are nonzero vectors. The set of $\alpha \in [0, 1]$ satisfying the inequality (3.4) must be not empty which follows from the application of Proposition 2.3 to $X = B(z, r)$ at step 4.

Note that conditions $0 \leq \alpha \leq 1$ and $(s_1 - s_2)\alpha + s_2 \geq 0$ are equivalent to conditions $\alpha^- \leq \alpha \leq \alpha^+$ where

$$\alpha^- = \begin{cases} \max\{0, \frac{-s_2}{s_1 - s_2}\}, & \text{if } s_1 > s_2, \\ 0, & \text{if } s_1 \leq s_2, \end{cases} \quad \text{and} \quad \alpha^+ = \begin{cases} \min\{1, \frac{-s_2}{s_1 - s_2}\}, & \text{if } s_1 < s_2, \\ 1, & \text{if } s_1 \geq s_2. \end{cases}$$

Thus in place of solving (3.4) with respect to $\alpha \in [0, 1]$ we may equivalently solve the system

$$\begin{cases} ((s_1 - s_2)\alpha + s_2)^2 \geq r^2 \|(h_1 - h_2)\alpha + h_2\|^2, \\ \alpha^- \leq \alpha \leq \alpha^+. \end{cases} \tag{3.5}$$

The first inequality is obtained from (3.4) by taking squares of both sides. The correctness of this operation is ensured by the condition $\alpha \in [\alpha^-, \alpha^+]$ because it implies $(s_1 - s_2)\alpha + s_2 \geq 0$.

Since

$$\|(h_1 - h_2)\alpha + h_2\|^2 = ((h_1 - h_2)\alpha + h_2)((h_1 - h_2)\alpha + h_2)^T,$$

it follows that the first inequality in (3.5) can be written in the form

$$\begin{aligned} &((s_1 - s_2)^2 - r^2(h_1 - h_2)(h_1 - h_2)^T) \cdot \alpha^2 + \\ &\qquad 2 \cdot ((s_1 - s_2)s_2 - r^2(h_1 - h_2)h_2^T) \cdot \alpha + s_2^2 - r^2 h_2 h_2^T \geq 0. \end{aligned}$$

Let us denote the left-hand side of this inequality by $f(\alpha)$ where $f$ is the corresponding quadratic function. Maximizing $f$ subject to the conditions $\alpha^- \leq \alpha \leq \alpha^+$, we obtain a value of $\alpha$ giving a solution to (3.5). The maximum of $f$ is delivered either by one of the bounds $\alpha^-$ or $\alpha^+$ or by a zero of the derivative of $f$. Differentiating by $\alpha$, we obtain

$$\frac{\partial f}{\partial \alpha}(\alpha) = 2((s_1 - s_2)^2 - r^2(h_1 - h_2)(h_1 - h_2)^T) \cdot \alpha + 2((s_1 - s_2)s_2 - r^2(h_1 - h_2)h_2^T). \tag{3.6}$$

If the coefficient of $\alpha$ is zero in the formula (3.6), then $f$ is a linear function and, to obtain a solution for (3.5), we choose one of the values $\alpha^-$ or $\alpha^+$. Otherwise, the derivative is equal

to zero at the point

$$\alpha_0 = \frac{r^2(h_1 - h_2)h_2^T - (s_1 - s_2)s_2}{(s_1 - s_2)^2 - r^2(h_1 - h_2)(h_1 - h_2)^T}. \tag{3.7}$$

In this case we should choose among $\alpha^-$, $\alpha^+$, and $\alpha_0$. The value $\alpha_0$ is taken into account only if $\alpha_0 \in [\alpha^-, \alpha^+]$. Note that the value $\alpha = \frac{-s_2}{s_1 - s_2}$ will never be chosen because it turns the expression $(s_1 - s_2)\alpha + s_2$ into zero which would imply $\|(h_1 - h_2)\alpha + h_2\| = 0$ by (3.5). But the length of $(h_1 - h_2)\alpha + h_2$ cannot be zero which follows from $h_1 \neq -\gamma h_2$ for all $\gamma > 0$ and the fact that $h_1$ and $h_2$ are nonzero vectors. We conclude that the value $\alpha$ found by the above single-variable optimization may be equal only to one of the values 0, 1, or $\alpha_0$. This gives an $O(n)$ algorithm for computing $\alpha$ by explicit formulas.

**Running time and correctness of the algorithm.** The PRELIMINARY D&C ALGORITHM either fails or solves the task $(z, r, \varepsilon)$. This is proved by the fact that the ELEMENTARY PROCEDURE, if the radius is sufficiently small, returns a correct solution. Assuming inductively that recursive calls return correct solutions to the respective tasks, we conclude that step 4 is correct. The depth of recursion is bounded by $\left\lceil \log_{1+\theta} \frac{2r\|c_{\max}\|}{\varepsilon} \right\rceil$ because the condition under which step 1 returns a solution is $\hat{r} \leq \frac{\varepsilon}{2\|c_{\max}\|}$ and the recursive calls at steps 2 and 3 are performed with respect to the current radius $\hat{r}$ divided by $1 + \theta$. Since at most two recursive calls are performed by each iteration, the number of iterations is bounded by

$$2^{\left\lceil \log_{1+\theta} \frac{2r\|c_{\max}\|}{\varepsilon} \right\rceil + 1} = O\left( \left( \frac{r\|c_{\max}\|}{\varepsilon} \right)^{\frac{1}{\log_2(1+\theta)}} \right).$$

It seems to be hard to obtain an analogous estimate of the sizes of the numbers which occur during the computation. This is the reason why we modify the PRELIMINARY D&C ALGORITHM in Section 3.2 below.

## 3.2 Divide-and-conquer algorithm

The divide-and-conquer algorithm, which we will call the D&C ALGORITHM, basically preserves the structure of its preliminary version (Algorithm 3.1). One of the main differences is that the modified algorithm rounds the components of the centers of the balls to integer

multiples of

$$\mu = \frac{\theta}{1+\theta} \cdot \frac{\varepsilon}{4 \cdot n \cdot c_{\max} c_{\max}^T}$$

so that the sizes are polynomially bounded in the size of the system and in $\frac{r\|c_{\max}\|}{\varepsilon}$.

We assume that both the initial radius $r$ and the parameter $\mu$ are given explicitly in the form of pairs $(\mu', \mu'')$ and $(r', r'')$ of positive integers. That is, $\mu = \frac{\mu'}{\mu''}$ and $r = \frac{r'}{r''}$. We can choose $\mu' = \theta' \cdot \varepsilon'$ and $\mu'' = 4 \cdot \varepsilon'' \cdot (\theta' + \theta'') \cdot n \cdot c_{\max} c_{\max}^T$, where the pair $(\varepsilon', \varepsilon'')$ of integers represents $\varepsilon$.

We also assume that a positive integer number $\rho$ is known such that $|z_j| \leq \rho$ and the product $\rho z_j$ is integer for every component $z_j$ of the initial center $z$.

Let us define

$$\eta = (\rho \cdot \det\, AA^T \cdot \mu'')^2.$$

Note that the matrix $(\det\, AA^T) \cdot (AA^T)^{-1}$ is integer because $A$ is integer. The multiplier $\eta$ will be used at step 1 to guarantee that any inequality returned by step 1 has integer coefficients.

We have the following algorithm:

**Algorithm 3.2** D&C ALGORITHM

**Input:** $(z, r, \varepsilon)$.

**Output:** Either $x^*$ with (i) or $(h, \delta)$ with (ii).

**1. if** $r \leq \frac{\varepsilon}{2\|c_{\max}\|}$ **then**

Solve $(z, r, \varepsilon)$ by the ELEMENTARY PROCEDURE.

**if** the procedure returns $x^*$ **then** return $x^*$.

**if** the procedure returns $h = c_i$ and $\delta = d_i$ **then** return $(h, \delta)$.

**if** the procedure returns $h = (z - p(z))^T$ and $\delta = hp(z)$ **then**

Set $h := \eta(z - p(z))^T$ and $\delta := hp(z)$ and return $(h, \delta)$.

**end if**

**end if**

**2.** Run the D&C ALGORITHM with the input $(z, \frac{1}{1+\theta} r, \varepsilon)$.

**if** the recursive call returns $x^*$ with (i) **then** return $x^*$;

**else** let $(h_1, \delta_1)$ be returned by the recursive call.

**if** $\frac{h_1 z - \delta_1}{\|h_1\|} \geq r$ **then** return $(h, \delta) = (h_1, \delta_1)$.

**3.** $z^0 := z - \frac{h_1 z - \delta_1}{h_1 h_1^T} h_1^T$.

$z' := (z'_1, \ldots, z'_n)^T$, $z'_j = \mu \left\lfloor \frac{z^0_j}{\mu} \right\rfloor$, $j = 1, \ldots, n$.

Run the D&C ALGORITHM with the input $(z', \frac{1}{1+\theta} r + n\mu, \varepsilon)$.

**if** the recursive call returns $x^*$ with (i) **then** return $x^*$;

**else** let $(h_2, \delta_2)$ be returned by the recursive call.

**if** $\frac{h_2 z - \delta_2}{\|h_2\|} \geq r$ **then** return $(h, \delta) = (h_2, \delta_2)$.

**4. if** $h_1 = -\gamma h_2$ for some $\gamma > 0$ **then** STOP. The algorithm fails.

**else**

Set $h = \tilde{\alpha}' h_1 + (\tilde{\alpha}'' - \tilde{\alpha}') h_2$ and $\delta = \tilde{\alpha}' \delta_1 + (\tilde{\alpha}'' - \tilde{\alpha}') \delta_2$.

($\tilde{\alpha}'$ and $\tilde{\alpha}''$ are integers such that $hx > \delta$ for all $x$ in $B(z, r)$.)

Return $(h, \delta)$.

**end if**

**Remark.** *If $h_1 x \leq \delta_1$ is an induced inequality and at the same time $\frac{h_1 z - \delta_1}{\|h_1\|} \geq r$, then $h_1 x > \delta_1$ for all $x \in B(z, r)$. The inequality $h_1 x \leq \delta_1$ is a solution of the task $(z, r, \varepsilon)$ therefore. In the same way, if $h_2 x \leq \delta_2$ is an induced inequality and $\frac{h_2 z - \delta_2}{\|h_2\|} \geq r$, then $h_2 x \leq \delta_2$ is a solution of the task $(z, r, \varepsilon)$. So before proceeding to the next step the algorithm performs additional comparisons $\frac{h_1 z - \delta_1}{\|h_1\|} \geq r$ and $\frac{h_2 z - \delta_2}{\|h_2\|} \geq r$ at steps 2 and 3, respectively.*

Steps 1 and 4 of the D&C ALGORITHM differ from those of the PRELIMINARY D&C ALGORITHM by additional multiplications by integers which are chosen so that to guarantee that the induced inequalities are represented by integer vectors of coefficients. Step 3 performs a rounding of $z^0$. The rounding may shift $z^0$ to a point $z'$. The recursive call at step 3 is performed with respect to $z'$. To compensate the rounding, the algorithm chooses a larger radius for the recursive call at step 3 than the one for the recursive call at step 2.

To consider step 3 in detail, let us denote the current task by $(\hat{z}, \hat{r}, \varepsilon)$. The point $z^0 = \hat{z} - \frac{h_1 \hat{z} - \delta_1}{h_1 h_1^T} h_1^T$ is the projection of $\hat{z}$ onto the half-space $\{x | h_1 x \leq \delta_1\}$. (The induced inequality

$h_1 x \leq \delta_1$ is returned by the recursive call at step 2.) This point is rounded so that the components of the obtained vector $z'$ become multiples of $\mu$. That is, $z' = (z'_1, \ldots, z'_n)^T$ where $z'_j = \mu \left\lfloor \frac{z^0_j}{\mu} \right\rfloor$, $j = 1, \ldots, n$. The recursive call at step 3 is performed with respect to the radius $\frac{1}{1+\theta}\hat{r} + n\mu$. The item $n\mu$ compensates the rounding. Indeed, the inclusion

$$B\left(z^0, \frac{1}{1+\theta}\hat{r}\right) \subseteq B\left(z', \frac{1}{1+\theta}\hat{r} + n\mu\right) \tag{3.8}$$

follows from the fact that if

$$\|x - z^0\| < \frac{1}{1+\theta}\hat{r},$$

then

$$\|x - z'\| = \|x - z^0 + z^0 - z'\| \leq \|x - z^0\| + \|z^0 - z'\| < \frac{1}{1+\theta}\hat{r} + \sqrt{n}\mu < \frac{1}{1+\theta}\hat{r} + n\mu.$$

The inclusion (3.8), in combination with the inclusion (3.2) which applies to $\hat{z}$ and $\hat{r}$, gives

$$B(\hat{z}, \hat{r}) \cap \{x | h_1 x \leq \delta_1\} \subseteq B\left(z', \frac{1}{1+\theta}\hat{r} + n\mu\right). \tag{3.9}$$

Note that $\hat{r} > \frac{\varepsilon}{2\|c_{\max}\|}$ because otherwise step 1 uses the ELEMENTARY PROCEDURE. Therefore, the choice of $\mu$ guarantees that

$$\frac{1}{1+\theta}\hat{r} + n\mu < \hat{r} \tag{3.10}$$

because

$$\frac{1}{1+\theta}\hat{r} + n\mu = \frac{1}{1+\theta}\hat{r} + \frac{\theta}{1+\theta} \cdot \frac{\varepsilon}{4c_{\max}c^T_{\max}} \leq \frac{1}{1+\theta}\hat{r} + \frac{\theta}{1+\theta} \cdot \frac{\varepsilon}{4\|c_{\max}\|} < \frac{1}{1+\theta}\hat{r} + \frac{\theta}{1+\theta}\hat{r} = \hat{r}.$$

(We use the fact that $c_{\max}$ is a nonzero integer vector which implies $c_{\max}c^T_{\max} \geq 1$ and therefore $c_{\max}c^T_{\max} \geq \|c_{\max}\|$.)

The inclusion (3.9) plays the same role as the inclusion (3.2) for the PRELIMINARY D&C ALGORITHM. So the recursive call at step 3 is performed with respect to $\left(z', \frac{1}{1+\theta}\hat{r} + n\mu\right)$. The radius for this recursive call is larger by $n\mu$ than that at step 2 and smaller than the current radius $\hat{r}$, as follows from (3.10).

We will now explain an iteration which is applied to a current input $(\hat{z}, \hat{r}, \varepsilon)$. Note that for every iteration the current radius is represented in the form $\hat{r} = \frac{\hat{r}'}{\hat{r}''}$ where $\hat{r}'$ and $\hat{r}''$ are

positive integers. This means that such representation of the corresponding radius must be calculated before every recursive call.

**Step 1.** Under the same conditions as for the preliminary version, return $h = \eta(\hat{z} - p(\hat{z}))^T$ and $\delta = hp(\hat{z})$ in place of $h = (\hat{z} - p(\hat{z}))^T$ and $\delta = hp(\hat{z})$.

The multiplication by $\eta$ ensures that the components of $(h, \delta)$ are integers. (See Proposition 3.1 below.) The obtained inequality is equivalent to the corresponding inequality $hx \leq \delta$ at step 1 of the PRELIMINARY D&C ALGORITHM.

**Step 2.** Note that $\frac{1}{1+\theta} = \frac{\theta''}{\theta' + \theta''}$. Represent the radius for the recursive call as

$$\frac{1}{1+\theta}\hat{r} = \frac{\theta'' \cdot \hat{r}'}{(\theta' + \theta'') \cdot \hat{r}''}. \tag{3.11}$$

Run the D&C ALGORITHM with $(\hat{z}, \frac{1}{1+\theta}\hat{r}, \varepsilon)$. If the recursive call returns an $\varepsilon$-approximate solution $x^*$, then return $x^*$. Otherwise, let an induced inequality $h_1 x \leq \delta_1$ be returned by the recursive call. If $\frac{h_1\hat{z} - \delta_1}{\|h_1\|} \geq \hat{r}$, then return $(h, \delta) = (h_1, \delta_1)$. Else go to step 3.

**Step 3.** Calculate $z'$ by the rounding of $z^0 = \hat{z} - \frac{h_1\hat{z} - \delta_1}{h_1 h_1^T} h_1^T$. Represent the radius for the recursive call as

$$\frac{1}{1+\theta}\hat{r} + n\mu = \frac{\theta'' \cdot \hat{r}' \cdot \mu'' + n \cdot \mu' \cdot \hat{r}'' \cdot (\theta' + \theta'')}{(\theta' + \theta'') \cdot \hat{r}'' \cdot \mu''}. \tag{3.12}$$

Run the D&C ALGORITHM with $(z', \frac{1}{1+\theta}\hat{r} + n\mu, \varepsilon)$. If the recursive call returns an $\varepsilon$-approximate solution $x^*$, then return $x^*$. Otherwise, let an induced inequality $h_2 x \leq \delta_2$ be returned by the recursive call. If $\frac{h_2\hat{z} - \delta_2}{\|h_2\|} \geq \hat{r}$, then return $(h, \delta) = (h_2, \delta_2)$. Else go to step 4.

**Step 4.** At this step, it is assumed that the recursive calls at steps 2 and 3 have returned a correct output. That is, the induced inequalities $h_1 x \leq \delta_1$ and $h_2 x \leq \delta_2$ are such that $h_1 x > \delta_1$ for all $x \in B(\hat{z}, \frac{1}{1+\theta}\hat{r})$ and $h_2 x > \delta_2$ for all $x \in B(z', \frac{1}{1+\theta}\hat{r} + n\mu)$. If $h_1 = -\gamma h_2$ for some $\gamma > 0$, then the algorithm fails. In this case it stops. Otherwise, it finds an induced inequality $hx \leq \delta$ such that $hx > \delta$ for all $x \in B(\hat{z}, \hat{r})$ using the following observation. Since $h_2 x > \delta_2$ for all $x \in B\left(z', \frac{1}{1+\theta}\hat{r} + n\mu\right)$, the inclusion (3.9) implies that $h_2 x > \delta_2$ for all those points $x$ in $B(\hat{z}, \hat{r})$ which satisfy the inequality $h_1 x \leq \delta_1$. Then Proposition 2.3, applied to $X = B(\hat{z}, \hat{r})$, claims that there exists $\alpha \in [0, 1]$ such that

$$(\alpha h_1 + (1-\alpha)h_2)x > \alpha\delta_1 + (1-\alpha)\delta_2$$

for all $x \in B(\hat{z}, \hat{r})$. The inequality

$$(\alpha h_1 + (1-\alpha)h_2)x \le \alpha\delta_1 + (1-\alpha)\delta_2 \tag{3.13}$$

is induced by the system because $h_1 x \le \delta_1$ and $h_2 x \le \delta_2$ are induced by the system. Even if $(h_1, \delta_1)$ and $(h_2, \delta_2)$ are integer vectors, this inequality may have fractional coefficients. To produce an integer output, the algorithm constructs positive integer values $\tilde{\alpha}'$ and $\tilde{\alpha}''$ and obtains an equivalent induced inequality

$$(\tilde{\alpha}' h_1 + (\tilde{\alpha}'' - \tilde{\alpha}')h_2)x \le \tilde{\alpha}'\delta_1 + (\tilde{\alpha}'' - \tilde{\alpha}')\delta_2$$

whose coefficients are integers if $(h_1, \delta_1)$ and $(h_2, \delta_2)$ are integer vectors.

The following procedure calculates the appropriate integer values $\tilde{\alpha}'$ and $\tilde{\alpha}''$. At first, the value $\alpha$ is computed by the same formulas as for the preliminary version. That is, $\alpha$ takes one of the values 0, 1, or $\alpha_0$. With respect to the current task $(\hat{z}, \hat{r}, \varepsilon)$, the formula (3.7) should be rewritten as

$$\alpha_0 = \frac{\hat{r}^2(h_1 - h_2)h_2^T - (s_1 - s_2)s_2}{(s_1 - s_2)^2 - \hat{r}^2(h_1 - h_2)(h_1 - h_2)^T}.$$

where $s_1 = h_1\hat{z} - \delta_1$ and $s_2 = h_2\hat{z} - \delta_2$. If $\alpha = \alpha_0$, then

$$\alpha = \frac{\alpha'}{\alpha''} \tag{3.14}$$

where

$$\alpha' = \hat{r}^2(h_1 - h_2)h_2^T - (s_1 - s_2)s_2$$

and

$$\alpha'' = (s_1 - s_2)^2 - \hat{r}^2(h_1 - h_2)(h_1 - h_2)^T.$$

If $\alpha = 0$, then we put $\alpha' = 0$ and $\alpha'' = 1$. If $\alpha = 1$, then $\alpha' = 1$ and $\alpha'' = 1$.

**Remark.** *The rounding of $z^0$ at step 3 implies that any center that occurs during the execution of the algorithm is either the initial center $z$ or such that the components are integer multiples of $\mu$.*

The current center $\hat{z}$ is either the initial center $z$ or all components of $\hat{z}$ are integer multiples of $\mu$. We can use this observation for the construction of $\tilde{\alpha}'$ and $\tilde{\alpha}''$. Indeed, recalling the definition of $\rho$, define

$$\lambda = (\mu'' \cdot \hat{r}'' \cdot \rho)^2 \tag{3.15}$$

Calculate $\tilde{\alpha}'$ and $\tilde{\alpha}''$ as $\tilde{\alpha}' = \lambda \cdot \alpha'$ and $\tilde{\alpha}'' = \lambda \cdot \alpha''$. The values $\tilde{\alpha}'$ and $\tilde{\alpha}''$ are integers if $(h_1, \delta_1)$ and $(h_2, \delta_2)$ are integer vectors, which follows from the above remark and from the fact that $\hat{r} = \frac{\hat{r}'}{\hat{r}''}$.

Denote $h = \tilde{\alpha}'h_1 + (\tilde{\alpha}'' - \tilde{\alpha}')h_2$ and $\delta = \tilde{\alpha}'\delta_1 + (\tilde{\alpha}'' - \tilde{\alpha}')\delta_2$. The inequality $hx \leq \delta$ is equivalent to (3.13) because $hx \leq \delta$ is obtained from (3.13) by multiplying by the positive value $\lambda\alpha''$. Then $hx \leq \delta$ is a solution of the task $(\hat{z}, \hat{r}, \varepsilon)$.

So we have described step 4. This step requires $O(n)$ arithmetic operations, which follows from the explicit formulas for computing $\alpha$.

**Proposition 3.1** *In the course of the algorithm, $h_1$ and $h_2$ are integer vectors and $\delta_1$ and $\delta_2$ are integer numbers.*

**Proof.** If $(h, \delta) = (c_i, d_i)$ is returned by step 1, then $(h, \delta)$ is an integer vector because $(c_i, d_i)$ is an integer vector. The current center $\hat{z}$ is either the initial center $z$ or such that the components of $\hat{z}$ are integer multiplies of $\mu$. The matrix $(\det AA^T) \cdot (AA^T)^{-1}$ is integer because $A$ is integer. It follows that if step 1 returns $h = \eta(z - p(z))$ and $\delta = hp(z)$ then $(h, \delta)$ is integer because of the multiplication by $\eta$. At step 4, $(h, \delta)$ is computed in such a way that if the components of $(h_1, \delta_1)$ and $(h_2, \delta_2)$ are integers then the components of $(h, \delta)$ are integers as well. ∎

**Remark.** *The structure of recursive calls is representable in the form of a binary tree. The nodes in this tree correspond to iterations performed by the algorithm. The root corresponds to the first iteration. The length of the path from the root to the node representing an iteration is called the level of recursion (of this iteration). The height of the tree is called the depth of recursion. We denote it by $L$. Levels of recursion take the integer values from 0 to $L$. The level 0 corresponds to the first iteration which is the iteration applied to the initial task $(z, r, \varepsilon)$.*

**Lemma 3.1** *The* D&C ALGORITHM *either fails or solves the task* $(z, r, \varepsilon)$ *in*

$$O\left(\left(\frac{r\|c_{\max}\|}{\varepsilon}\right)^{\frac{1}{\log_2(1+\theta)}}\right)$$

*iterations.*

**Proof.** Consider an iteration with the current input $(\hat{z}, \hat{r}, \varepsilon)$. Let $l \geq 1$ be the current level of recursion. We can prove by induction that

$$\hat{r} \leq \frac{1}{(1+\theta)^l} r + \sum_{q=0}^{l-1} \frac{1}{(1+\theta)^q} n\mu.$$

Indeed if $l = 1$ then we are done. Consider the case $l \geq 2$. Let $\bar{r}$ be the radius at the iteration which performs the recursive call corresponding to the current iteration. Then the current radius $\hat{r}$ is equal to either $\frac{1}{1+\theta}\bar{r}$ or $\frac{1}{1+\theta}\bar{r} + \mu n$. Taking into account that the level of recursion of the mentioned iteration is equal to $l - 1$, we assume by induction that

$$\bar{r} \leq \frac{1}{(1+\theta)^{l-1}} r + \sum_{q=0}^{l-2} \frac{1}{(1+\theta)^q} n\mu.$$

Then

$$\hat{r} \leq \frac{1}{1+\theta}\bar{r} + n\mu \leq \frac{1}{(1+\theta)^l} r + \sum_{q=1}^{l-1} \frac{1}{(1+\theta)^q} n\mu + n\mu = \frac{1}{(1+\theta)^l} r + \sum_{q=0}^{l-1} \frac{1}{(1+\theta)^q} n\mu.$$

This proves the upper bound for $\hat{r}$. Taking into account

$$\sum_{q=0}^{l-1} \frac{1}{(1+\theta)^q} \leq \sum_{q=0}^{\infty} \frac{1}{(1+\theta)^q} = \frac{1+\theta}{\theta},$$

we obtain

$$\hat{r} \leq \frac{1}{(1+\theta)^l} r + \frac{1+\theta}{\theta} n\mu \leq \frac{1}{(1+\theta)^l} r + \frac{\varepsilon}{4\|c_{\max}\|}$$

because $\|c_{\max}\| \leq c_{\max} c_{\max}^T$. It follows that if $l$ is sufficiently large to ensure the inequality

$$\frac{1}{(1+\theta)^l} r + \frac{\varepsilon}{4\|c_{\max}\|} \leq \frac{\varepsilon}{2\|c_{\max}\|},$$

then $\hat{r} \leq \frac{\varepsilon}{2\|c_{\max}\|}$ and step 1 returns a solution to the current task. Thus the depth of recursion is bounded by a value of $l$ at which the above inequality holds true. That is, we have

$$L \leq \left\lceil \log_{1+\theta} \frac{4r\|c_{\max}\|}{\varepsilon} \right\rceil.$$

Since at most two recursive calls are performed at every iteration, it follows that the total number of iterations does not exceed

$$2^{\lceil \log_{1+\theta} \frac{4r\|c_{\max}\|}{\varepsilon} \rceil + 1} = O\left(\left(\frac{r\|c_{\max}\|}{\varepsilon}\right)^{\frac{1}{\log_2(1+\theta)}}\right).$$

(Recall that $\theta$ is constant.) Thus, we have proved that the algorithm is finite. The same argument as for the PRELIMINARY ALGORITHM proves that step 1 produces a correct solution for the elementary case. Step 4 works correctly if the recursive calls produce correct solutions. This proves by induction that the algorithm either fails, in the sense that we have already discussed, or solves the task. ∎

## 3.3 Growth of the numbers

We concentrate now on the analysis of absolute values of the integers occurring in the computation. At first let us find out how far the centers calculated in the course of the algorithm can be from the initial center $z$.

**Proposition 3.2** *At every iteration of the* D&C ALGORITHM *the current center $\hat{z}$ satisfies $\|\hat{z} - z\| \leq L(r + n\mu)$ where $L$ is the depth of recursion.*

**Proof.** Let $(\hat{z}, \hat{r}, \varepsilon)$ be the current input. The first recursive call (the one at step 2) uses the same center $\hat{z}$. The center $z'$ for the second recursive call (the one at step 3) is calculated only if $\frac{h_1\hat{z}-\delta_1}{\|h_1\|} < \hat{r}$ because otherwise step 2 returns $(h_1, \delta_1)$ as a solution of the current task. Note that $h_1\hat{z} > \delta_1$. The value $\frac{h_1\hat{z}-\delta_1}{\|h_1\|}$ is the distance between $\hat{z}$ and the hyperplane $\{x|h_1x = \delta_1\}$. This distance is less than $\hat{r}$. Since $z^0$ is the projection of $\hat{z}$ onto $\{x|h_1x = \delta_1\}$,

$$\|z^0 - \hat{z}\| < \hat{r} \leq r.$$

($\hat{r} \leq r$ because the radii decrease with increasing level of recursion. This follows from (3.10).) The distance between $z^0$ and $z'$ is not greater than $n\mu$. Hence

$$\|z' - \hat{z}\| \leq r + n\mu.$$

Denoting by $l$ the current level of recursion, let us assume that

$$\|\hat{z} - z\| \leq l(r + n\mu).$$

Then

$$\|z' - z\| = \|z' - \hat{z} + \hat{z} - z\| \leq \|z' - \hat{z}\| + \|\hat{z} - z\| \leq (l+1)(r + n\mu).$$

For the iteration corresponding to the second recursive call, $z'$ plays the role of the current center. By induction on the levels of recursion, we conclude that the distance between any center $\hat{z}$ occurring in the computation and the initial center $z$ never exceeds $L(r + n\mu)$ because the levels of recursion are bounded by $L$. ∎

Let $(\hat{z}, \hat{r}, \varepsilon)$ be the current input considered by some iteration. Denoting

$$R = L \left\lceil \frac{r + n\mu}{\mu} \right\rceil,$$

and taking into account $\mu' \geq 2$ and $\mu'' \geq 2$, by the above proposition we get

$$|\hat{z}_j| \leq |z_j| + R\mu \leq \rho + R\mu = \frac{\rho\mu'' + R\mu'}{\mu''} \leq \frac{\rho\mu'' R\mu'}{\mu''} = R \cdot \rho \cdot \mu' \tag{3.16}$$

for every component $\hat{z}_j$ of $\hat{z}$. Let $M(v)$ denote the maximum absolute value of the components of a vector $v$. For instance, $M(h, \delta)$ is the maximum absolute value of the components of the vector composed of $h$ and $\delta$. Recall that the current radius $\hat{r}$ is represented as $\hat{r} = \frac{\hat{r}'}{\hat{r}''}$, where $\hat{r}'$ and $\hat{r}''$ are positive integers. Denote

$$\tau = (R \cdot \rho \cdot \mu' \cdot \hat{r}')^2. \tag{3.17}$$

**Proposition 3.3** *The coefficients $\tilde{\alpha}'$ and $\tilde{\alpha}''$ calculated at step 4 are positive integers which are bounded by*

$$20 \cdot n^2 \cdot \lambda \cdot \tau \cdot \left(M(h_1, h_2, \delta_1, \delta_2)\right)^2.$$

**Proof.** Step 4 constructs $\tilde{\alpha}' = \lambda \cdot \alpha'$ and $\tilde{\alpha}'' = \lambda \cdot \alpha''$. The values $s_1$ and $s_2$ in the formula for computing $\alpha_0$ should be defined as $s_1 = h_1\hat{z} - \delta_1$ and $s_2 = h_2\hat{z} - \delta_2$. To shorten notation, we write $M$ in place of $M(h_1, h_2, \delta_1, \delta_2)$. Taking into account (3.16), we have

$$|s_1| \leq (n+1) \cdot R \cdot \rho \cdot \mu' \cdot M \leq 2n \cdot \sqrt{\tau} \cdot M.$$

In the same way,

$$|s_2| \le 2n \cdot \sqrt{\tau} \cdot M.$$

Assume that step 4 computes $\alpha = \alpha_0$. In this case

$$\tilde{\alpha}' = \lambda \cdot (\hat{r}^2(h_1 - h_2)h_2^T - (s_1 - s_2)s_2)$$

and

$$\tilde{\alpha}'' = \lambda \cdot ((s_1 - s_2)^2 - \hat{r}^2(h_1 - h_2)(h_1 - h_2)^T).$$

These values are integers because of the multiplication by $\lambda$ and because $(h_1, \delta_1)$ and $(h_2, \delta_2)$ are integer vectors by Proposition 3.1. The values $|h_1 h_2^T|$, $h_1 h_1^T$, and $h_2 h_2^T$ are bounded by $n \cdot M^2$, and the values $|s_1 s_2|$, $s_1^2$, and $s_2^2$ are bounded by $4n^2 \cdot \tau \cdot M^2$. Taking into account that $\hat{r} = \frac{\hat{r}'}{\hat{r}''} \le \hat{r}'$ because $\hat{r}''$ is integer, we obtain

$$\begin{aligned}
\tilde{\alpha}'' &\le \lambda \cdot (s_1^2 + 2|s_1 s_2| + s_2^2 + (\hat{r}')^2 \cdot (h_1 h_1^T + 2|h_1 h_2^T| + h_2 h_2^T)) \\
&\le \lambda \cdot (16n^2 \cdot \tau \cdot M^2 + (\hat{r}')^2 \cdot (4n \cdot M^2)) \\
&\le \lambda \cdot (16n^2 \cdot \tau \cdot M^2 + 4n^2 \cdot \tau \cdot M^2) = 20 \cdot n^2 \cdot \lambda \cdot \tau \cdot M^2.
\end{aligned}$$

The latter value is an upper bound also for $\tilde{\alpha}'$. The same estimate is true for the cases $\alpha = 0$ and $\alpha = 1$. ∎

The values $\lambda$ and $\tau$ depend on $\hat{r}'$ and $\hat{r}''$. We evaluate now upper bounds $\lambda_{\max}$ and $\tau_{\max}$ which depend only on the initial data.

The formulas (3.11) and (3.12) give the representation of the current radius $\hat{r}$ in the form $\hat{r} = \frac{\hat{r}'}{\hat{r}''}$ where $\hat{r}'$ and $\hat{r}''$ are positive integers. Using these formulas, by induction on levels of recursion we can prove that

$$\hat{r}' \le (n \cdot \theta' \cdot \theta'' \cdot \mu' \cdot \mu'' \cdot r' \cdot r'')^{3L^2}$$

and

$$\hat{r}'' \le (\theta' \cdot \theta'' \cdot \mu'')^L \cdot r''.$$

(Recall that $L$ is the depth of recursion and the integers $r'$ and $r''$ represent the initial radius $r$. We can obtain the above upper bounds as follows. Taking into account that the coefficients

are positive integers and $\theta' \geq 2$ and $\theta'' \geq 2$, we replace the addition by multiplication in (3.11) and (3.12). At first we derive the bound for $\hat{r}''$, by induction on levels of recursion, and then use this bound to derive the bound for $\hat{r}'$.) Denoting

$$\phi = (n \cdot \theta' \cdot \theta'' \cdot \mu' \cdot \mu'' \cdot r' \cdot r'')^{3L^2},$$

we obtain that $\hat{r}' \leq \phi$ and $\hat{r}'' \leq \phi$. Let us define

$$\lambda_{\max} = (\mu'' \cdot \phi \cdot \rho)^2$$

and

$$\tau_{\max} = (R \cdot \rho \cdot \mu' \cdot \phi)^2.$$

So we have $\lambda \leq \lambda_{\max}$ and $\tau \leq \tau_{\max}$ in the course of the algorithm. (See the definitions (3.15) and (3.17) of $\lambda$ and $\tau$.)

Let $p_{\max}$ be the maximum absolute value of

- $|\det AA^T|$
- the elements of $C$ and $d$,
- the elements of $|\det AA^T| \cdot A^T(AA^T)^{-1}A$, and
- the elements of $|\det AA^T| \cdot A^T(AA^T)^{-1}b$.

The determinant $\det AA^T$ is integer because $A$ is integer. The matrix $|\det AA^T| \cdot A^T(AA^T)^{-1}A$ and the vector $|\det AA^T| \cdot A^T(AA^T)^{-1}b$ are integer because of the multiplication by $|\det AA^T|$. So the number $p_{\max}$ is a positive integer. Let us denote

$$\zeta = 60 \cdot n^3 \cdot \lambda_{\max} \cdot (\tau_{\max})^2 \cdot (p_{\max})^4.$$

**Lemma 3.2** *Consider an iteration with the current input $(\hat{z}, \hat{r}, \varepsilon)$. Let $q$ be an upper bound on the difference between the maximum level of recursion achievable from the current iteration and the current level of recursion. Then*

$$M(h, \delta) \leq \zeta^{4^q}.$$

**Proof.** Let $q$ be equal to 0. In this case the current iteration finds a solution at step 1. That is, the condition $\hat{r} \leq \frac{\varepsilon}{2\|c_{\max}\|}$ holds. If the solution returned by step 1 is $(h, \delta) = (c_i, d_i)$, then $M(h, \delta) \leq p_{\max} \leq \zeta$. Assume now that step 1 returns $h = \eta(\hat{z} - p(\hat{z}))^T$ and $\delta = hp(\hat{z})$. Recall that

$$p(\hat{z}) = \hat{z} + A^T(AA^T)^{-1}(b - A\hat{z}).$$

By (3.16) we have

$$M(\hat{z}) \leq R \cdot \rho \cdot \mu'.$$

Note that

$$M\left(A^T(AA^T)^{-1}b\right) \leq p_{\max}$$

and

$$M\left(A^T(AA^T)^{-1}A\hat{z}\right) \leq n \cdot M\left(A^T(AA^T)^{-1}A\right) \cdot M(\hat{z}) \leq n \cdot p_{\max} \cdot R \cdot \rho \cdot \mu'.$$

It follows that

$$M(p(\hat{z})) \leq M(\hat{z}) + M\left(A^T(AA^T)^{-1}b\right) + M\left(A^T(AA^T)^{-1}A\hat{z}\right) \leq 2n \cdot R \cdot \rho \cdot \mu' \cdot p_{\max}.$$

Taking into account $\eta \leq \lambda_{\max} \cdot (p_{\max})^2$ which follows from $|\det AA^T| \leq p_{\max}$, in case $h = \eta(\hat{z} - p(\hat{z}))^T$ we obtain

$$M(h) \leq \eta \cdot (M(\hat{z}) + M(p(\hat{z}))) \leq \eta \cdot (3n \cdot R \cdot \rho \cdot \mu' \cdot p_{\max}) \leq 3n \cdot \lambda_{\max} \cdot \tau_{\max} \cdot (p_{\max})^3$$

and

$$|\delta| = |hp(\hat{z})| \leq n \cdot M(h) \cdot M(p(\hat{z})) \leq 6n^3 \cdot \lambda_{\max} \cdot (\tau_{\max})^2 \cdot (p_{\max})^4.$$

It follows that $M(h, \delta) \leq \zeta$ in the case $q = 0$.

Let $q \geq 1$. Let us assume by induction that $M(h_1, \delta_1) \leq \zeta^{4^{q-1}}$ and $M(h_2, \delta_2) \leq \zeta^{4^{q-1}}$. (The number $q - 1$ is an upper bound on the differences between the maximum levels of recursion which are achievable from the iterations that returned $(h_1, \delta_1)$ and $(h_2, \delta_2)$ and the level of recursion of these iterations.) Note that $M(h_1, h_2, \delta_1, \delta_2) \leq \zeta^{4^{q-1}}$. Let $(h, \delta)$ be returned by step 4. Since

$$(h, \delta) = (\tilde{\alpha}'h_1 + (\tilde{\alpha}'' - \tilde{\alpha}')h_2, \tilde{\alpha}'\delta_1 + (\tilde{\alpha}'' - \tilde{\alpha}')\delta_2),$$

it follows that

$$M(h,\delta) \le 3\max\{\tilde{\alpha}', \tilde{\alpha}''\} \cdot M(h_1, h_2, \delta_1, \delta_2)$$

The estimate in Proposition 3.3 yields

$$3\max\{\tilde{\alpha}', \tilde{\alpha}''\} \le \zeta \cdot (M(h_1, h_2, \delta_1, \delta_2))^2 \le \zeta \cdot \zeta^{2\cdot 4^{q-1}}.$$

Taking into account that $\zeta \le \zeta^{4^{q-1}}$ because $\zeta$ is a positive integer, we can write

$$M(h,\delta) \le \zeta \cdot \zeta^{2\cdot 4^{q-1}} \cdot \zeta^{4^{q-1}} \le \zeta^{4^{q-1}} \cdot \zeta^{2\cdot 4^{q-1}} \cdot \zeta^{4^{q-1}} = \zeta^{4^q}.$$

■

Since for two nonnegative integers $\omega_1$ and $\omega_2$ the inequality $\omega_1 \le \omega_2$ implies $\mathrm{size}(\omega_1) \le \mathrm{size}(\omega_2)$ where $\mathrm{size}(\cdot)$ denotes the size of a number, we have the following estimate of the size of $\zeta$:

**Proposition 3.4** *The size of $\zeta$ is bounded by a polynomial in the size of the system of inequalities, in $L$, and in the sizes of $\mu'$, $\mu''$, $r'$, $r''$, and $\rho$.*

**Proof.** Since the coefficients are integers,

$$R = L\left\lceil \frac{r}{\mu} + n \right\rceil = L\left\lceil \frac{r'\mu''}{r''\mu'} + n \right\rceil \le L\left\lceil r'\mu'' + n \right\rceil = L(r'\mu'' + n)$$

and

$$\mathrm{size}(R) \le \mathrm{size}(L) + \mathrm{size}(r') + \mathrm{size}(\mu'') + \mathrm{size}(n).$$

Using the formulas defining the corresponding coefficients, we obtain

$$\mathrm{size}(\phi) \le 3L^2 \cdot (\mathrm{size}(n) + \mathrm{size}(\theta') + \mathrm{size}(\theta'') + \mathrm{size}(\mu') + \mathrm{size}(\mu'') + \mathrm{size}(r') + \mathrm{size}(r'')),$$

$$\mathrm{size}(\lambda_{\max}) \le 2(\mathrm{size}(\mu'') + \mathrm{size}(\phi) + \mathrm{size}(\rho)),$$

and

$$\mathrm{size}(\tau_{\max}) \le 2(\mathrm{size}(\mu') + \mathrm{size}(\phi) + \mathrm{size}(\rho) + \mathrm{size}(R)).$$

Note that

$$\operatorname{size}(\zeta) \le \operatorname{size}(60) + 3 \cdot \operatorname{size}(n) + \operatorname{size}(\lambda_{\max}) + 2 \cdot \operatorname{size}(\tau_{\max}) + 4 \cdot \operatorname{size}(p_{\max}).$$

Then, because $L \ge 1$ and the number 24 is sufficiently large, it follows that

$$\operatorname{size}(\zeta) \le \operatorname{size}(60) + 4 \cdot \operatorname{size}(p_{\max}) + 24 \cdot L^2 \sum_{\omega \in \{n,\theta',\theta'',\mu',\mu'',r',r'',\rho,L\}} \operatorname{size}(\omega).$$

It remains to estimate the size of $p_{\max}$. Note that the sizes of $\det AA^T$ and $(AA^T)^{-1}$ are bounded by a polynomial in the size of $A$. Then the definition of $p_{\max}$ implies that the size of $p_{\max}$ is bounded by a polynomial in the size of the system of inequalities. Thus, the size of $\zeta$ is bounded by a polynomial in the size of the system, in $L$, and in the sizes of $\mu'$, $\mu''$, $r'$, $r''$, and $\rho$. Recall that $\theta'$ and $\theta''$ are constants. ∎

## 3.4 Running time and the sizes of the numbers

Let $K$ denote the number of iterations performed by the D&C ALGORITHM with respect to $(z, r, \varepsilon)$. Lemma 3.1 states that

$$K \le O\left(\left(\frac{r\|c_{\max}\|}{\varepsilon}\right)^{\frac{1}{\log_2(1+\theta)}}\right).$$

In the theorem below, by the size of the input we understand the size of the system plus the sizes of $r$, $z$, and $\varepsilon$. We assume that $A^T(AA^T)^{-1}b$, $A^T(AA^T)^{-1}A$, and $\eta$ are computed before step 1 of the first iteration. This computation is further called the preprocessing step.

**Theorem 3.1** *The* D&C ALGORITHM *either fails or solves the task* $(z, r, \varepsilon)$. *If equipped with the preprocessing step, the algorithm performs at most*

$$O\left(m^3 + m^2 n + n^2 m + K \cdot \left(n \log\left(\frac{\rho + L(r + n\mu)}{\mu}\right) + n^2 + N\right)\right)$$

*arithmetic operations. Here,* $L$ *is the depth of recursion and* $N$ *is the number of nonzero components of the matrix* $C$. *The sizes of the numbers occurring in the computation are bounded by a polynomial in the size of the input, in* $\frac{r\|c_{\max}\|}{\varepsilon}$, *and in the sizes of* $\mu'$, $\mu''$, $r'$, $r''$, *and* $\rho$.

**Proof.** The fact that the algorithm either fails or solves the task follows from Lemma 3.1.

After computing $AA^T$, the matrix $(AA^T)^{-1}$ is computable in $O(m^3)$ arithmetic operations on numbers of polynomial size. Note that, because of $\det AA^T$, the computation of $\eta$ also requires $O(m^3)$ arithmetic operations. (The computation of $(AA^T)^{-1}$ and $\det AA^T$ in strongly polynomial time needs an appropriate implementation of the Gaussian elimination for the case of integer coefficients. Such implementation was given, for instance, by Edmonds in [3].) Taking into account multiplications of the matrices, we see that the preprocessing step requires $O(m^3 + m^2n + n^2m)$ arithmetic operations.

Let us estimate the number of arithmetic operations required by each one of $K$ iterations.

To perform the comparison $\frac{c_iz-d_i}{\|c_i\|} < r$, it is sufficient to check the sign of $c_iz - d_i$ and compare $\frac{(c_iz-d_i)^2}{\|c_i\|^2}$ and $r^2$ taking into account that $\|c_i\|^2 = c_ic_i^T$. It follows that the comparisons $\frac{c_iz-d_i}{\|c_i\|} < r$, $i = 1, \ldots, k$, require $O(N)$ arithmetic operations provided that the nonzero entries of $C$ are organized as an appropriate data structure. So $O(n + N)$ is a bound for the whole number of arithmetic operations required by the comparisons in the ELEMENTARY PROCEDURE.

Since the preprocessing step makes it possible to compute projections in $O(n^2)$ arithmetic operations, the ELEMENTARY PROCEDURE runs in $O(n^2 + N)$ time.

The computation of $\alpha$, $\tilde{\alpha}'$, and $\tilde{\alpha}''$ by the explicit formulas requires $O(n)$ time. The verification whether $h_1 = -\gamma h_2$ for some $\gamma > 0$ takes $O(n)$ time.

The logarithm in the estimate corresponds to the calculation of $z'_j = \mu \left\lfloor \frac{z^0_j}{\mu} \right\rfloor$ which is performed by a binary search over multiples of $\mu$ and takes

$$O\left(\log\left(\frac{\rho + L(r + n\mu)}{\mu}\right)\right)$$

time because

$$|z'_j| \leq |z_j| + L(r + n\mu) \leq \rho + L(r + n\mu)$$

due to the fact that $z'$ belongs to $B(z, L(r + n\mu))$ by Proposition 3.2.

Note that the size of $x^*$ is bounded by a polynomial in the size of the input and in the sizes of $\mu$ and $\rho$.

To estimate the sizes of the inequalities, we use Lemma 3.2 from which it follows that

$$M(h,\delta) \leq \zeta^{4^L}$$

because we can choose $q \leq L$. This implies

$$\text{size}(M(h,\delta)) \leq 4^L \cdot \text{size}(\zeta)$$

because $M(h,\delta)$ and $\zeta^{4^L}$ are nonnegative integers. ($(h,\delta)$ is integer by Proposition 3.1. size$(\cdot)$ denotes the binary size of the corresponding object.) It follows that

$$\text{size}((h,\delta)) \leq (n+1) \cdot 4^L \cdot \text{size}(\zeta)$$

at every iteration. Since

$$L \leq \left\lceil \log_{1+\theta} \left( \frac{4r\|c_{\max}\|}{\varepsilon} \right) \right\rceil$$

by the proof of Lemma 3.1, we have

$$\text{size}((h,\delta)) \leq O\left( (n+1) \left( \frac{r\|c_{\max}\|}{\varepsilon} \right)^{\frac{2}{\log_2(1+\theta)}} \text{size}(\zeta) \right).$$

Proposition 3.4 claims that the size of $\zeta$ is bounded by a polynomial in the size of the system, in $L$, and in the sizes of $\mu'$, $\mu''$, $r'$, $r''$, and $\rho$.

Thus we can see that the sizes of the numbers occurring in the computation are bounded by a polynomial in the size of the input, in $\frac{r\|c_{\max}\|}{\varepsilon}$, and in the sizes of $\mu'$, $\mu''$, $r'$, $r''$, and $\rho$. ∎

The following corollary will be needed for maintaining induced inequalities in the form of linear combinations of the original constraints.

**Corollary 3.1** *The* D&C ALGORITHM *can be equipped with an additional procedure which for every iteration of the algorithm, if the iteration returns an induced inequality $hx \leq \delta$, represents $h$ and $\delta$ in the form $h = a + \sum_{i=1}^{k} \alpha_i c_i$ and $\delta = \xi + \sum_{i=1}^{k} \alpha_i d_i$, where $(a,\xi)$ is a linear combination of the rows of $(A|b)$ and $\alpha_i$ are nonnegative integers, and the sizes of $(a,\xi)$ and $\alpha_i$, $i = 1,\ldots,k$, are bounded by a polynomial in the size of the input, in $\frac{r\|c_{\max}\|}{\varepsilon}$, and in the sizes of $\mu'$, $\mu''$, $r'$, $r''$, and $\rho$. The estimate of the running time of the* D&C ALGORITHM *equipped with the procedure remains the same as in the above theorem.*

**Proof.** Consider an iteration with the current input $(\hat{z}, \hat{r}, \varepsilon)$. The procedure runs as follows. If step 1 returns $h = \eta(\hat{z} - p(\hat{z}))^T$ and $\delta = hp(\hat{z})$, then set $a := h$, $\xi := \delta$, and $\alpha_i := 0$ for all $i$ because the inequalities $Cx \leq d$ are not involved in $hx \leq \delta$. If step 1 returns $(h, \delta) = (c_l, d_l)$ for some $l = 1, \ldots, k$, then set $\alpha_l := 1$, $\alpha_i := 0$ for all $i \neq l$, and let $(a, \xi)$ be a zero vector.

Let the recursive calls at steps 2 and 3 return $h_1 x \leq \delta_1$ and $h_2 x \leq \delta_2$ where $h_1$ and $h_2$ are represented, by the procedure, as

$$h_1 = a' + \textstyle\sum_{i=1}^{k} \alpha'_i c_i \quad \text{and} \quad h_2 = a'' + \sum_{i=1}^{k} \alpha''_i c_i.$$

($\delta_1$ and $\delta_2$ are the corresponding combinations of the right-hand sides.) Step 4 calculates $h = \tilde{\alpha}' h_1 + (\tilde{\alpha}'' - \tilde{\alpha}')h_2$ and $\delta = \tilde{\alpha}'\delta_1 + (\tilde{\alpha}'' - \tilde{\alpha}')\delta_2$. Then the procedure sets

$$\alpha_i := \tilde{\alpha}'\alpha'_i + (\tilde{\alpha}'' - \tilde{\alpha}')\alpha''_i, \tag{3.18}$$

for all $i$, and also $a := h - \sum_{i=1}^{k} \alpha_i c_i$, and $\xi := \delta - \sum_{i=1}^{k} \alpha_i d_i$.

If step 1 produces a solution, then $\alpha_i$ is either 0 or 1, for all $i$. Assuming by induction that $\alpha'_i$ and $\alpha''_i$ are integers, we obtain that $\alpha_i$ are integers as well because $\tilde{\alpha}'$ and $\tilde{\alpha}''$ are integers. Let $\tilde{\alpha}_{\max}$ be the maximum value taken by $\tilde{\alpha}'$ and $\tilde{\alpha}''$ in the course of the algorithm. Note that $\tilde{\alpha}_{\max}$ is a positive integer because $\tilde{\alpha}'' > 0$. Taking into account that the coefficients are integers, from (3.18) we obtain

$$\text{size}(\alpha_i) \leq 3 \cdot \text{size}(\tilde{\alpha}_{\max}) + \text{size}(\alpha'_i) + \text{size}(\alpha''_i). \tag{3.19}$$

Let $q$ be an upper bound on the difference between the maximum level of recursion achievable from the current iteration and the current level of recursion. We will now prove that

$$\text{size}(\alpha_i) \leq 5^q \cdot \text{size}(\tilde{\alpha}_{\max}).$$

This holds for $q = 0$ because either $\alpha_i = 1$ or $\alpha_i = 0$ in this case. Let $q \geq 1$. Assume by induction that $\text{size}(\alpha'_i)$ and $\text{size}(\alpha''_i)$ are upper bounded by $5^{q-1} \cdot \text{size}(\tilde{\alpha}_{\max})$. Then (3.19) gives

$$\text{size}(\alpha_i) \leq 3 \cdot \text{size}(\tilde{\alpha}_{\max}) + 2 \cdot 5^{q-1} \cdot \text{size}(\tilde{\alpha}_{\max})$$

$$\leq 3 \cdot 5^{q-1} \cdot \text{size}(\tilde{\alpha}_{\max}) + 2 \cdot 5^{q-1} \cdot \text{size}(\tilde{\alpha}_{\max})$$

$$= 5^q \cdot \text{size}(\tilde{\alpha}_{\max}).$$

Since $q \leq L$, we get

$$\text{size}(\alpha_i) \leq O\left(\left(\frac{r\|c_{\max}\|}{\varepsilon}\right)^{\frac{\log_2 5}{\log_2(1+\theta)}} \text{size}(\tilde{\alpha}_{\max})\right).$$

Theorem 3.1 implies that $\text{size}(\tilde{\alpha}_{\max})$, as well as $\text{size}((h,\delta))$, is bounded by a polynomial in the size of the input, in $\frac{r\|c_{\max}\|}{\varepsilon}$, and in the sizes of $\mu'$, $\mu''$, $r'$, $r''$, and $\rho$. So we have the required estimate for the size of $\alpha_i$ and also for the size of $(a,\xi)$. ∎

# 4 Parametrization

Again, we consider the system

$$Ax = b, Cx \leq d.$$

The set of feasible solutions is denoted by

$$P = \{x \mid Ax = b, Cx \leq d\}.$$

We will apply the D&C ALGORITHM (Algorithm 3.2) not directly to the system in question, but to its parameterized version with strengthened constraints. By the parametrization we understand introducing a new variable $t \geq 1$ and multiplying by it the right-hand sides of the equations and inequalities.

## 4.1 Parameterized system

The parametrization by the new variable $t$ gives the following system:

$$\begin{aligned} Ax - bt &= \mathbf{0}, \\ Cx - dt &\leq \mathbf{0}, \\ -t &\leq -1. \end{aligned} \tag{4.1}$$

A solution of the system is a column vector of the form $(x^T|t)^T$. It is convenient to use the notation $(x;t)$.

The affine subspace is now given by $Ax - bt = \mathbf{0}$. If $A$ has full rank, then $(A| - b)$ also has full rank.

The system (4.1) is feasible if and only if the system $Ax = b, Cx \leq d$, is feasible. Indeed, let $(x^*; t^*)$ be a solution of (4.1). Then $A(\frac{1}{t^*}x^*) = b$ and $C(\frac{1}{t^*}x^*) \leq d$. (The division is correct because $t^*$ is positive.) That is, $\frac{1}{t^*}x^* \in P$. Conversely, if $x^* \in P$, then $(x^*; 1)$ is a solution of (4.1). The vector $(x^*; 1)$ is called the associated solution. This solution is uniquely determined by $x^*$.

Let $\varepsilon$ be a positive value. We apply the D&C ALGORITHM to the system

$$\begin{aligned} &Ax - bt = \mathbf{0}, \\ &Cx - dt \leq -\varepsilon\mathbf{1}, \\ &-t \leq -1 - \varepsilon. \end{aligned} \tag{4.2}$$

The system (4.2) may be infeasible even if the original parameterized system (4.1) is feasible because (4.2) is obtained from (4.1) by strengthening the inequalities. A solution of (4.1), if exists, is an $\varepsilon$-approximate solution to (4.2). Conversely, an $\varepsilon$-approximate solution to (4.2) delivers an exact solution for (4.1). (Recall that, by the definition in Subsection 2.2, an $\varepsilon$-approximate solution should exactly satisfy the equations and approximately satisfy the inequalities.) Of course, an $\varepsilon$-approximate solution may exist even if the system is infeasible.

With respect to the strengthened parameterized system (4.2), the original parameterized system (4.1) plays a role of an approximate system and therefore the task in Subsection 2.2 should be rewritten as follows:

Given $((x^0; t^0), r, \varepsilon)$, where $\varepsilon > 0$ and $r > 0$, either find an $\varepsilon$-approximate solution $(x^*; t^*)$ of the strengthened parameterized system (4.2), that is, $(x^*; t^*)$ such that

(I) $Ax^* - bt^* = \mathbf{0}$, $Cx^* - dt^* \leq \mathbf{0}$, $-t^* \leq -1$, ($(x^*; t^*)$ is a solution of the original parameterized system (4.1)),

or a nonzero row vector $h$ and a value $\delta$ such that

(II) $h(x; t) > \delta$ for all $(x; t) \in B((x^0; t^0), r)$ and the inequality $h(x; t) \leq \delta$ is induced by the strengthened parameterized system (4.2).

To solve the above task, we apply the D&C ALGORITHM to the strengthened parameterized system (4.2) with $((x^0;t^0),r,\varepsilon)$. Theorem 3.1 implies that if the D&C ALGORITHM does not fail or find an $\varepsilon$-approximate solution, then it returns an inequality $h(x;t)\leq\delta$ which is induced by the strengthened system (4.2). This inequality can be written as

$$a(x;t)+\sum_{i=1}^{k}\alpha_i(c_ix-d_it)-\beta t\leq-\sum_{i=1}^{k}\alpha_i\varepsilon-\beta(1+\varepsilon). \tag{4.3}$$

Here, $a$ is a linear combination of the rows of $(A|-b)$. (The coefficients in the combination determining $a$ may be negative.) The $\beta$ and the coefficients $\alpha_i$ are *nonnegative*. Throughout this section we will use the notation

$$h=a+\sum_{i=1}^{k}\alpha_i(c_i,-d_i)-\beta(\mathbf{0},1).$$

Here $\mathbf{0}$ denotes an all-zero row vector. The inequality (4.3) can be written as

$$h(x;t)\leq-\sum_{i=1}^{k}\alpha_i\varepsilon-\beta(1+\varepsilon).$$

We have

$$h(x^0;t^0)>-\sum_{i=1}^{k}\alpha_i\varepsilon-\beta(1+\varepsilon), \tag{4.4}$$

due to (II), which implies

**Proposition 4.1** *If $Ax^0-bt^0=\mathbf{0}$, then at least one of the coefficients $\alpha_i$, $i=1,\ldots,k$, or $\beta$ is nonzero.*

**Proof.** Assume by contradiction that the mentioned coefficients are zero. Then $h=a$. This implies $h(x^0;t^0)=0$. Then we have a contradiction because (4.4) implies $0>0$. ∎

## 4.2 The case when the algorithm fails to find a solution

Let the algorithm fail and $(h_1,\delta_1)$ and $(h_2,\delta_2)$ be returned by the recursive calls at the iteration where the algorithm fails. Since $h_1(x;t)\leq\delta_1$ and $h_2(x;t)\leq\delta_2$ are induced by (4.2), it follows that, for some linear combinations $a'$ and $a''$ of the rows of $(A|-b)$,

$$h_1=a'+\sum_{i=1}^{k}\alpha_i'(c_i,-d_i)-\beta'(\mathbf{0},1),$$

and

$$h_2 = a'' + \sum_{i=1}^{k} \alpha_i''(c_i, -d_i) - \beta''(\mathbf{0}, 1),$$

where $\alpha_i' \geq 0$, $\alpha_i'' \geq 0$, $\beta' \geq 0$, and $\beta'' \geq 0$. The values $\delta_1$ and $\delta_2$ are linear combinations of the right-hand sides of (4.2) with the same coefficients:

$$\delta_1 = -\sum_{i=1}^{k} \alpha_i'\varepsilon - \beta'(1+\varepsilon)$$

and

$$\delta_2 = -\sum_{i=1}^{k} \alpha_i''\varepsilon - \beta''(1+\varepsilon).$$

In this section, the vectors $h_1$ and $h_2$ and the values $\delta_1$ and $\delta_2$ are supposed to have the above form.

The algorithm fails only if $h_1 = -\gamma h_2$ where $\gamma > 0$. (The algorithm never fails at the deepest levels of recursion.) In this case we have

$$a' + \gamma a'' = -\sum_{i=1}^{k} \alpha_i'(c_i, -d_i) + \beta'(\mathbf{0}, 1) - \gamma \sum_{i=1}^{k} \alpha_i''(c_i, -d_i) + \gamma\beta''(\mathbf{0}, 1). \tag{4.5}$$

The lemma below uses this equation to show that if the D&C ALGORITHM fails, then certain properties of solutions of the system $Ax = b, Cx \leq d$, can be derived by analyzing the coefficients $\alpha_i'$, $\alpha_i''$, $\beta'$, and $\beta''$. In particular, if $\beta' > 0$ and $\beta'' > 0$ then the lemma implies that the system $Ax = b, Cx \leq d$, is infeasible. If at least one of the coefficients $\alpha_l'$ or $\alpha_l''$ is positive, then $c_l x = d_l$ for all $x \in P$, which allows us to increase the number of equations and reduce the number of inequalities.

**Lemma 4.1** *Let the algorithm fail. Let the system $Ax = b, Cx \leq d$, be feasible. Then*

*1) $\beta' = 0$ and $\beta'' = 0$.*

*2) At least one of the coefficients $\alpha_i'$, $\alpha_i''$, $i = 1, \ldots, k$, is nonzero. If $\alpha_l' > 0$ or $\alpha_l'' > 0$, then $c_l x = d_l$ for all $x$ satisfying $Ax = b, Cx \leq d$.*

**Proof.** 1) Since the system $Ax = b, Cx \leq d$, is feasible, it has a solution $x^*$. The associated solution $(x^*; 1)$ satisfies (4.1). Taking into account the minus signs on the left of the sums

in the equation (4.5), we see that the scalar products of both sides of (4.5) with $(x^*; 1)$ give

$$0 = \sum_{i=1}^{k} \alpha_i'(d_i - c_i x^*) + \beta' + \gamma \sum_{i=1}^{k} \alpha_i''(d_i - c_i x^*) + \gamma\beta'' \tag{4.6}$$

because $a'(x^*; 1) = 0$ and $a''(x^*; 1) = 0$. Since the values $d_i - c_i x^*$ are nonnegative because of $Cx^* \leq d$, and the coefficients $\alpha_i'$, $\alpha_i''$, $\beta'$, and $\beta''$ are nonnegative, and $\gamma > 0$, it follows that $\beta' = 0$ and $\beta'' = 0$ because otherwise the right-hand side of the above equation would be positive.

2) So we have $\beta' = 0$ and $\beta'' = 0$. The algorithm implies that there exists $l$ such that $\alpha_l' > 0$ or $\alpha_l'' > 0$. (Otherwise, $\delta_1 = \delta_2 = 0$. Then $h_2 z^0 = \delta_2$ because $0 = \delta_1 = h_1 z^0 = -\gamma h_2 z^0$. But $h_2 z^0 > \delta_2$ due to (3.8).) Let us assume that there is $x^*$ in $P$ such that $c_l x^* < d_l$. The associated solution $(x^*; 1)$ is feasible for (4.1). It follows that the scalar product of both sides of (4.5) with $(x^*; 1)$ gives the equation (4.6). The values $d_i - c_i x^*$ are nonnegative and the coefficients $\alpha_i'$, $\alpha_i''$, $\beta'$, and $\beta''$ are also nonnegative, and $\gamma > 0$. Then, since $d_l - c_l x^* > 0$, it follows from (4.6) that $\alpha_l' = \alpha_l'' = 0$. We have a contradiction because $\alpha_l' > 0$ or $\alpha_l'' > 0$. Our assumption that there is $x^* \in P$ with $c_l x^* < d_l$ is false therefore. It follows that $c_l x = d_l$ for all $x$ such that $Ax = b$, $Cx \leq d$. ∎

## 4.3 The case when the algorithm returns an induced inequality

We now assume that the algorithm returns the induced inequality $hx \leq \delta$. It has the form (4.3). Denote the respective hyperplane by

$$H = \left\{ (x; t) \mid h(x; t) = -\sum_{i=1}^{k} \alpha_i \varepsilon - \beta(1 + \varepsilon) \right\}.$$

Lemma 4.2, which will be given later on, implies the following facts, provided that $(x^0; t^0)$, $r$, and $\varepsilon$ are chosen appropriately. Assume there is no solution to $Ax = b, Cx \leq d$, with $\|x\| \geq r^*$. If $\beta \geq \max_i \alpha_i$, then the system is infeasible. If $\alpha_l = \max_i \alpha_i$ and $\alpha_l \geq \beta$, then $c_l x \geq d_l - \frac{1}{2}$ holds for all solutions $x$ of the system.

Let us suppose that the system $Ax = b, Cx \leq d$, has a solution $x^*$. The associated

solution $(x^*;1)$ is feasible for (4.1). Additionally, we assume that

$$\sum_{i=1}^{k} \alpha_i(d_i - c_i x^*) + \beta \neq 0$$

which is necessary to define $\Delta = \lambda^*(x^*;1)$ with

$$\lambda^* = -\frac{\sum_{i=1}^{k} \alpha_i \varepsilon + \beta\varepsilon}{\sum_{i=1}^{k} \alpha_i(d_i - c_i x^*) + \beta}.$$

The crucial role in the proof of Lemma 4.2 belongs to the inequality

$$\|(x;t) - (x^0;t^0)\| \geq r - \|\Delta\| \tag{4.7}$$

which, as it will be proved, holds for every solution $(x;t)$ of the original parameterized system (4.1).

Note that we are free to choose any solution $x^*$, satisfying the above conditions, to construct $\Delta$. The estimate gives a lower bound for the distance between a solution $(x;t)$ and the initial center $(x^0;t^0)$.

**Proof of the inequality (4.7).** Taking into account $a(x^*;1) = 0$, we get $a\Delta = 0$. It follows that

$$h\Delta = \lambda^* \left( \sum_{i=1}^{k} \alpha_i(c_i x^* - d_i) - \beta \right) = \sum_{i=1}^{k} \alpha_i \varepsilon + \beta\varepsilon.$$

Since the value

$$h(x^0;t^0) + \sum_{i=1}^{k} \alpha_i \varepsilon + \beta(1+\varepsilon)$$

is positive by (4.4), the distance between $(x^0;t^0)$ and $H$ is given by

$$\frac{h(x^0;t^0) + \sum_{i=1}^{k} \alpha_i \varepsilon + \beta(1+\varepsilon)}{\|h\|} = \frac{h(x^0;t^0) + \beta + h\Delta}{\|h\|} \leq$$

$$\leq \frac{h(x^0;t^0) + \beta + \|h\| \cdot \|\Delta\|}{\|h\|} = \frac{h(x^0;t^0) + \beta}{\|h\|} + \|\Delta\|.$$

(We use the Cauchy-Schwarz inequality.) The distance between $(x^0;t^0)$ and $H$ is not less than $r$ due to (II). Then it follows from the above relations that

$$\frac{h(x^0;t^0) + \beta}{\|h\|} \geq \frac{h(x^0;t^0) + \sum_{i=1}^{k} \alpha_i \varepsilon + \beta(1+\varepsilon)}{\|h\|} - \|\Delta\| \geq r - \|\Delta\|. \tag{4.8}$$

The inequality (4.7) is obvious if $r \leq \|\Delta\|$. Let us consider the case $r > \|\Delta\|$. Then the inequalities (4.8) imply

$$\frac{h(x^0;t^0)+\beta}{\|h\|} > 0.$$

It follows that

$$h(x^0;t^0) > -\beta.$$

Note that the inequality

$$h(x;t) \leq -\beta$$

is induced by the original parameterized system (4.1). (The only nonzero component in the right-hand side of the system (4.1) is equal to $-1$ and corresponds to the right-hand side of the inequality $-t \leq -1$. Multiplied by $\beta$, it gives $-\beta$.) Every solution $(x^*;t^*)$ of the system (4.1) satisfies $h(x^*;t^*) \leq -\beta$ therefore. It follows that $(x^0;t^0)$ and $(x^*;t^*)$ are separated by the hyperplane $\{(x;t)|h(x;t) = -\beta\}$. Then the distance between $(x^0;t^0)$ and $(x^*;t^*)$ is not less than the distance between $(x^0;t^0)$ and the hyperplane $\{(x;t)|h(x;t) = -\beta\}$. Since this distance is given by the value $\frac{h(x^0;t^0)+\beta}{\|h\|}$ which is not less than $r - \|\Delta\|$ due to (4.8), we get the inequality (4.7) for all solutions $(x;t)$ of the original parameterized system (4.1). ∎

Now we are ready to formulate the lemma.

**Lemma 4.2** *Let $r^*$ be a positive value. Let the algorithm return the induced inequality $hx \leq \delta$ when applied to (4.2) with the input parameters $(x^0;t^0) = (\mathbf{0};0)$, $r = (2k+3)\cdot(r^*+1)$, and $\varepsilon = 1$.*

*1) If $\beta \geq \max_i \alpha_i$, then $\|x\| \geq r^*$ for all $x$ in $P$.*

*2) If $\alpha_l = \max_i \alpha_i$, and $\alpha_l \geq \beta$, then $d_l - c_l x \leq \frac{1}{2}$ for all $x$ in $P$ such that $\|x\| < r^*$.*

**Proof.** Since $Ax^0 - bt^0 = \mathbf{0}$, by Proposition 4.1 at least one of the coefficients $\beta$, $\alpha_i$, $i = 1,\ldots,k$, is nonzero. It follows that if $\beta$ is maximum among the coefficients then $\beta > 0$, and if $\alpha_l$ is maximum then $\alpha_l > 0$.

1) $\beta \geq \max_i \alpha_i$. In this case, $\beta > 0$. Assume there exists $x^* \in P$ with $\|x^*\| < r^*$. Let $(x^*; 1)$ be the associated solution of (4.1). Since $\alpha_i \geq 0$ and $d_i - c_i x^* \geq 0$ for all $i$, we obtain

$$\sum_{i=1}^{k} \alpha_i (d_i - c_i x^*) + \beta \geq \beta > 0.$$

Taking into account $\varepsilon = 1$ and $\beta \geq \alpha_i$ for all $i$, we get

$$|\lambda^*| = \frac{\sum_{i=1}^{k} \alpha_i \varepsilon + \beta\varepsilon}{\sum_{i=1}^{k} \alpha_i (d_i - c_i x^*) + \beta} \leq \frac{(k+1)\beta}{\beta} = k + 1.$$

Then

$$\|\Delta\| = |\lambda^*| \cdot \|(x^*; 1)\| \leq (k+1)\|(x^*; 1)\| \leq (k+1)(\|x^*\| + 1) < (k+1)(r^* + 1).$$

By the inequality (4.7), applied to $(x^0; t^0) = (\mathbf{0}; 0)$ and $r = (2k+3)(r^*+1)$, this implies that

$$\|(x; t)\| \geq (k+2)(r^* + 1)$$

for every solution $(x; t)$ of the original parameterized system (4.1). Then, since $(x^*; 1)$ is a solution of (4.1), it follows that

$$\|(x^*; 1)\| \geq (k+2)(r^* + 1) \geq r^* + 1.$$

Hence $\|x^*\| \geq r^*$ due to $\|(x^*; 1)\| \leq \|x^*\| + 1$. We have a contradiction because we assumed that $\|x^*\| < r^*$.

2) $\alpha_l = \max_i \alpha_i$, and $\alpha_l \geq \beta$. In this case, $\alpha_l > 0$ by Proposition 4.1. Let there exist $x^*$ in $P$ such that $\|x^*\| < r^*$. Assume that $d_l - c_l x^* > \frac{1}{2}$. Since all coefficients $\alpha_i$, $\beta$, and all values $d_i - c_i x^*$ are nonnegative, we obtain

$$\sum_{i=1}^{k} \alpha_i (d_i - c_i x^*) + \beta \geq \alpha_l (d_l - c_l x^*) > 0.$$

Taking into account $\varepsilon = 1$ we get

$$|\lambda^*| = \frac{\sum_{i=1}^{k} \alpha_i \varepsilon + \beta\varepsilon}{\sum_{i=1}^{k} \alpha_i (d_i - c_i x^*) + \beta} \leq \frac{(k+1)\alpha_l}{\alpha_l (d_l - c_l x^*)} = \frac{k+1}{d_l - c_l x^*} < 2(k+1).$$

We obtain

$$\|\Delta\| \leq 2(k+1)\|(x^*; 1)\| \leq 2(k+1)(\|x^*\| + 1) < 2(k+1)(r^* + 1)$$

Applying the inequality (4.7) to $(x^0; t^0) = (\mathbf{0}; 0)$ and $r = (2k+3)(r^*+1)$, we conclude that

$$\|(x;t)\| \geq r^* + 1$$

for every solution $(x;t)$ of (4.1). Then, since $(x^*;1)$ is a solution of (4.1), it follows that

$$\|(x^*;1)\| \geq r^* + 1.$$

Hence $\|x^*\| \geq r^*$. We have a contradiction because $\|x^*\| < r^*$. Thus our assumption, that $d_l - c_l x^* > \frac{1}{2}$, is false. Then the inequality $d_l - c_l x \leq \frac{1}{2}$ holds for all $x$ in $P$ with $\|x\| < r^*$. ∎

# 5 Polynomial algorithm

We now propose an algorithm which either finds a solution for the system $Ax = b, Cx \leq d$, or correctly decides that it has no integer solution. This algorithm runs in strongly polynomial time for systems of the form $Ax = b, \mathbf{0} \leq x \leq \mathbf{1}$.

The algorithm is a straightforward application of Lemmas 4.1 and 4.2. So $hx \leq \delta$ must be represented in the form (4.3). The coefficients $\alpha_i$, $i = 1, \ldots, k$, and $\beta$ can be calculated by the procedure which is given by Corollary 3.1.

Let $r^*$ be a positive value and all $x$ in $P$ satisfy $\|x\| < r^*$. We apply the D&C ALGORITHM to the strengthened parameterized system (4.2) with the following input parameters:

$$\begin{aligned} &(x^0; t^0) = (\mathbf{0}; 0), \\ &r = (2k+3)(r^*+1), \\ &\varepsilon = 1. \end{aligned} \tag{5.1}$$

These are exactly the same parameters as in Lemma 4.2.

If the D&C ALGORITHM returns an induced inequality, we use Lemma 4.2 as follows. (We take into account that $\|x\| < r^*$ for all $x \in P$.) If $\beta$ is the maximum coefficient, then part 1 of Lemma 4.2 implies that the system $Ax = b, Cx \leq d$, has no solutions because otherwise $\|x\| \geq r^*$ for all $x$ in $P$. If $\alpha_l$ is a maximum coefficient, then part 2 of Lemma 4.2 implies $d_l - \frac{1}{2} \leq c_l x$ for all $x$ in $P$. Then the equation

$$c_l x = d_l$$

holds for all *integer* solutions $x$ in $P$ because at the same time $c_l x \leq d_l$ and the coefficients of $C$ and $d$ are integers. This means that we may replace $c_l x \leq d_l$ by $c_l x = d_l$ without loss of any integer feasible solution. The repeated application of the above argument leads to the following algorithm:

**Algorithm 5.1** LP ALGORITHM

**Input:** $Ax = b$, $Cx \leq d$, and $r^*$ such that $\|x\| < r^*$ for all $x$ in $P$.
**Output:** Either $x^*$ in $P$ or the decision that there are no integer solutions.
**while** the set of inequalities $Cx \leq d$ is not empty
  **if** the system of equations $Ax = b$ has no solutions
    return "NO INTEGER SOLUTIONS."
  **else**
    If necessary, reduce $Ax = b$ to an equivalent system of full rank.
  **end if**
  Run the D&C ALGORITHM for the strengthened parameterized system (4.2)
  and the input parameters (5.1).
  **if** the D&C ALGORITHM returns an $\varepsilon$-approximate solution $(x', t')$
  **then** return $x^* = \frac{1}{t'}x'$.
  **if** the D&C ALGORITHM fails **then**
    Let the inequalities $h_1 x \leq \delta_1$ and $h_2 x \leq \delta_2$ be returned
    by the recursive calls at the iteration where the algorithm fails.
    (See Subsection 4.2.)
    Apply Lemma 4.1:
    **if** $\alpha_i', \alpha_i'' = 0$, $i = 1, \ldots, k$, **then** return "NO INTEGER SOLUTIONS."
    Find $l$ such that either $\alpha_l' > 0$ or $\alpha_l'' > 0$.
    Add $c_l x = d_l$ to $Ax = b$ and remove $c_l x \leq d_l$ from $Cx \leq d$.
  **else**
    Let $hx \leq \delta$ be returned by the D&C ALGORITHM.
    (See Subsection 4.3.)

Apply Lemma 4.2:

**if** $\beta \geq \max_i \alpha_i$ **then** return "NO INTEGER SOLUTIONS."

Find $l$ such that $\alpha_l = \max_i \alpha_i$.

Add $c_l x = d_l$ to $Ax = b$ and remove $c_l x \leq d_l$ from $Cx \leq d$.

**end if**

**end while**

**if** $Ax = b, Cx = d$, has no solutions **then** return "NO INTEGER SOLUTIONS."

**else** return a solution $x^*$ of the current system $Ax = b, Cx = d$.

Note that at the beginning of every iteration of the loop we run the Gaussian elimination for the subsystem formed by the equations. During each iteration of the loop, the LP ALGORITHM modifies the original system $Ax = b, Cx \leq d$, by replacing some inequality $c_l x \leq d_l$ by the equation $c_l x = d_l$. The current system has the form

$$\begin{aligned} &Ax = b, \\ &c_i x = d_i, \ \forall i \in S, \\ &c_i x \leq d_i, \ \forall i \in \{1, \ldots, k\} \setminus S, \end{aligned}$$

where $S$ is the set of indices of the inequalities in $Cx \leq d$ which have been replaced by the corresponding equations. Every solution $x$ of the current system satisfies $\|x\| < r^*$ because the same holds true for all feasible solutions of the original system. Lemmas 4.1 and 4.2 imply therefore that the set of integer feasible solutions of the current system is the same as that of the original system. A feasible solution of the current system is feasible also for the original system $Ax = b, Cx \leq d$. If the current system has no integer solutions, then the original system has no integer solutions either. The loop may run until the current system takes the form $Ax = b, Cx = d$. This system can be solved by an appropriate implementation of the Gaussian elimination in strongly polynomial time.

**Theorem 5.1** *Let the inequalities $Cx \leq d$ take the form $\mathbf{0} \leq x \leq \mathbf{1}$. Let $r^* = \left\lceil \sqrt{n+1} \right\rceil$. Then the* LP ALGORITHM *either finds a solution $x^*$ for $Ax = b, \mathbf{0} \leq x \leq \mathbf{1}$, or correctly decides that this system has no integer solutions. The algorithm runs in strongly polynomial time bounded by $O\left(n^{3+\frac{3}{2\log_2(1+\theta)}}\right)$.*

**Proof.** We have $\|x\| < \left\lceil \sqrt{n+1} \right\rceil = r^*$ for all solutions of $Ax = b, \mathbf{0} \le x \le \mathbf{1}$. Lemmas 4.1 and 4.2 prove that the LP ALGORITHM either finds a solution or decides that there is no integer solution for the system $Ax = b, \mathbf{0} \le x \le \mathbf{1}$. Note that $\|c_{\max}\| = \sqrt{2}$ because $c_{\max}$ refers now to the inequalities of the system (4.2). The structure of the input parameters (5.1) implies that

$$r = (4n+3)\left(\left\lceil \sqrt{n+1} \right\rceil + 1\right).$$

To apply the D&C ALGORITHM we need to calculate the values $\mu'$, $\mu''$, $r'$, $r''$, and $\rho$. Noting that $\varepsilon = 1$ and $c_{\max} c_{\max}^T = 2$, we set $\mu' = \theta'$ and $\mu'' = 8n(\theta' + \theta'')$. The value $\left\lceil \sqrt{n+1} \right\rceil$ is computable in $O(\log n)$ time by a binary search. The radius $r$ is integer. We set $r' = r$ and $r'' = 1$. Since the initial center $(x^0; t^0)$ is equal to $(\mathbf{0}; 0)$, we can set $\rho = 1$. Thus, the calculation of $\mu'$, $\mu''$, $r'$, $r''$, and $\rho$ requires $O(\log n)$ time.

So we have

$$\frac{\|c_{\max}\| r}{\varepsilon} \le 2(4n+3)\left(\left\lceil \sqrt{n+1} \right\rceil + 1\right) = O\left(n^{\frac{3}{2}}\right)$$

for every call of the D&C ALGORITHM. Then for every call of the D&C ALGORITHM Theorem 3.1 implies a strongly polynomial number of arithmetic operations on numbers of polynomial size. Taking into account that the number of such calls does not exceed $n$ and suppressing the constant multipliers in the estimate of the running time given by Theorem 3.1, we obtain the required estimate. (It also includes the running time of the Gaussian elimination.) ∎

The running time of the LP ALGORITHM is independent of $A$ and $b$. Recall that Tardos [7] developed a strongly polynomial algorithm, for linear optimization problems, whose running time is independent of the right-hand side and the objective function.

**Remark.** *The estimate* $O\left(n^{3+\frac{3}{2\log_2(1+\theta)}}\right)$ *tends to* $O(n^6)$ *as* $1+\theta$ *tends to* $\sqrt{2}$.

The inequalities $\mathbf{0} \le x \le \mathbf{1}$ impose bounds on variables. A bound is tight if it is achieved by some solution of the system. The condition of tightness is slightly more general than the condition of non-redundancy of inequalities.

**Corollary 5.1** *If the bounds on variables are tight in* $Ax = b, \mathbf{0} \le x \le \mathbf{1}$, *then the system is solvable in* $O\left(n^{2+\frac{3}{2\log_2(1+\theta)}}\right)$ *time. The condition of tightness can be replaced by the weaker*

*condition that for every $j$ there are feasible solutions $x^-$ and $x^+$ such that $x^-_j < \frac{1}{2}$ and $x^+_j > \frac{1}{2}$.*

**Proof.** If the system is feasible, then a solution will be found by the first iteration of the loop in the LP ALGORITHM. Indeed, if the D&C ALGORITHM fails, then there is an index $l$ such that one of the equations $x_l = 1$ or $x_l = 0$ holds for all feasible solutions. If the D&C ALGORITHM returns an induced inequality, then there is an index $l$ such that one of the inequalities $x_l \leq \frac{1}{2}$ or $x_l \geq \frac{1}{2}$ holds for every feasible solution of the system. That is, one of the bounds 1 or 0 is not tight. We have a contradiction. ∎

Note that the above corollary extends to systems of the form $Ax = b, u^- \leq x \leq u^+$, because they easily reduce to $Ax = b, \mathbf{0} \leq x \leq \mathbf{1}$.

The LP ALGORITHM can be used as a basic procedure in branch-and-bound algorithms for integer linear programming. If the LP ALGORITHM answers that there are no integer solutions for a subproblem, then the corresponding node in the branch-and-bound tree can be fathomed. Note that a usual linear solver would always find a solution if there is a fractional one, while the LP ALGORITHM may return the negative answer. Moreover, independently of the answer returned by the LP ALGORITHM, each iteration of the loop, if it is not the final one, allows the branch-and-bound algorithm to fix a value of the variable corresponding to the inequality which is replaced by the equation.

Note that the running time of the branch-and-bound algorithm based on the LP ALGORITHM does not depend on $A$ and $b$ because the LP ALGORITHM runs in strongly polynomial time.

The LP ALGORITHM can also be used by branch-and-bound algorithms for optimization problems. Let $c$ be an integer row vector and $\min\{cx | Ax = b, x \in \{0,1\}^n\}$ be a subproblem considered by a branch-and-bound algorithm. Let $\xi$ be the best known objective value for the original integer linear problem and $\xi - \gamma - 1$ be the lower bound on the optimal value of the current subproblem. Let us introduce a variable $v$ and add the constraints

$$cx + \gamma v = \xi - 1,\ 0 \leq v \leq 1,$$

to the system $Ax = b, \mathbf{0} \leq x \leq \mathbf{1}$. If the first iteration of the loop produces a solution, it

can be used for branching by the branch-and-bound algorithm. If the first iteration of the loop of the LP ALGORITHM fixes the value of a variable $x_j$, this reduces the search space. If the first iteration fixes the value of $v$, this, by the proof of Corollary 5.1, means that one of the bounds on the optimal value is not tight. Then, depending on what bound has been detected to be not tight, $\xi - \frac{1}{2}\gamma - 1$ is either a lower or an upper bound on those objective values of feasible solutions of the subproblem which are less than $\xi$.

In conclusion, let us note that a system $Ax = b, \mathbf{0} \leq x \leq u$, where $u$ is an integer column vector, can be transformed in polynomial time into a system $\tilde{A}\tilde{x} = \tilde{b}, \mathbf{0} \leq \tilde{x} \leq \mathbf{1}$, which has a 0,1-solution if and only if the original system is feasible. That is, the LP ALGORITHM implies a polynomial algorithm for linear programming [2].

## Acknowledgement

I wish to thank the anonymous referee and Kurt Anstreicher for pointing out errors in the early versions of the paper and making suggestions for the improvement of the structure and presentation.

## References

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

[2] Chubanov, S. 2010. A polynomial relaxation-type algorithm for linear programming. http://www.optimization-online.org/DB_HTML/2011/02/2915.html

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

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

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

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

[7] Tardos, E. 1986. A strongly polynomial algorithm to solve combinatorial linear programs. *Operations Research* **34**, 250-256.

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