ISSN 0965-5425, Computational Mathematics and Mathematical Physics, 2012, Vol. 52, No. 1, pp. 146–156. © Pleiades Publishing, Ltd., 2012.
Original Russian Text © N.Yu. Zolotykh, 2012, published in Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki, 2012, Vol. 52, No. 1, pp. 153–163.

New Modification of the Double Description Method for Constructing the Skeleton of a Polyhedral Cone

N. Yu. Zolotykh

Nizhni Novgorod State University, pr. Gagarina 23, Nizhni Novgorod, 603950 Russia

e-mail: Nikolai.Zolotykh@gmail.com

Received March 1, 2011; in final form, July 27, 2011

Abstract—A new modification of the double description method is proposed for constructing the skeleton of a polyhedral cone. Theoretical results and a numerical experiment show that the modification is considerably superior to the original algorithm in terms of speed.

DOI: 10.1134/S0965542512010162

Keywords: polyhedron, polyhedral cone, skeleton of a cone, convex hull, double description method.

INTRODUCTION

It is well known that each polyhedron in $\mathbb{R}^d$ can be represented in either of the following two ways:

(i) as the solution set of a system of linear inequalities;

(ii) as the sum (in the sense of Minkowski) of the conical hull of a set of vectors $u_1, u_2, \ldots, u_n$ and the convex hull of a set of points $v_1, v_2, \ldots, v_k$ in $\mathbb{R}^d$.

If the polyhedron is a solid figure, then the irreducible system of linear inequalities for it is uniquely defined up to the multiplication of each inequality by positive constants, and each inequality is associated with a facet (i.e., a face of maximum dimension). If the polyhedron does not contain nonzero affine subspaces, then the irreducible generating sets of vectors $u_1, u_2, \ldots, u_n$ and points $v_1, v_2, \ldots, v_k$ are uniquely determined up to the multiplication of $u_1, u_2, \ldots, u_n$ by positive scalars. Here, $v_1, v_2, \ldots, v_k$ are the vertices of the polyhedron and $u_1, u_2, \ldots, u_n$ are its extreme recession rays. Given representation (i), the construction of irreducible representation (ii) is called, loosely speaking, the *problem of finding a vertex description*. The inverse problem is one of *finding a facet description* or the *construction of a convex hull*. According to a classical theorem of Weyl, either of these two problems can be reduced to its inverse (dual) in at most linear time.

Similarly, every polyhedral cone in $\mathbb{R}^d$ can be represented in either of the following two ways:

(i) as the solution set of a homogeneous set of linear inequalities;

(ii) as the set of all nonnegative combinations of a set of vectors in $\mathbb{R}^d$.

The *skeleton* of a cone is defined as an irreducible system of vectors all of whose nonnegative combinations comprise that cone. If a cone is acute (i.e., contains no nonzero subspaces), then its skeleton is uniquely defined up to the multiplication of the vectors by positive scalars and forms the set of extreme rays of the cone.

There is a standard method for reducing the construction of a vertex/facet description for polyhedra to the corresponding problem for polyhedral cones. For example, a vertex description of a polyhedron $\{x \in \mathbb{R}^d : Ax \geq b\}$ can be found by solving a similar problem for the cone $\{(x, x_{d+1}) \in \mathbb{R}^{d+1} : Ax - bx_{d+1} \geq 0, x_{d+1} \geq 0\}$ and, then, setting $x_{d+1} = 1$.

These problems arise in many applications, such as computer graphics, physical simulation, image processing, cartography, computational biology, theoretical physicist, and others. The construction of a

Delaunay triangulation and a Voronoi diagram [1] can be reduced to the construction of a facet description (convex hull).

At present, it is not known whether there are algorithms solving these problems in a time that is a polynomial in the total length of the input and output. Moreover, none of the algorithms available has this property [2]. On the other hand, these problems arise in many applications and fast practical algorithms are necessary for their solution.

Below, we consider the classical double description method [3] (also known as the Motzkin–Burger algorithm), which solves the problems in question. Various improvements of this method were proposed, for example, in [4–13].

The double description method is "incremental," and its underlying idea can be described as follows. Consider the problem of constructing a vertex description of a polyhedron $\{x : Ax \le b\}$. First, the problem is solved for a subsystem of the system $Ax \le b$ (for example, for a single inequality or a subsystem of rank d). Then the inequalities of the original system are added to this subsystem one after another and the vertex description is updated each time. The name of the double description method is explained by the fact that each iteration involves two descriptions of the current polyhedron (a vertex and a facet one), and the other necessary information (e.g., the set of all edges of the current polyhedron) is calculated from them. Suppose that the inequality $ax \le \beta$ is added at the current iteration. The vertex set of a new polyhedron consists of vertices of the current one whose coordinates satisfy this inequality and the intersection points of its edges with the hyperplane $ax = \beta$.

The double description method does not update the complete lattice of faces in the current polyhedron, in contrast to other incremental algorithms, such as the beneath-beyond method [1] or the Shevchenko–Chirkov algorithm [12], and does not use triangulations, in contrast to [13–15], etc. Note that the size of the complete lattice of faces and the size of a triangulation can depend superpolynomially on the total size of the input and output (see [2]). For this reason, the double description method is frequently faster than other algorithms, for example, as applied to extremely degenerate problems. (The problem of finding a vertex description is degenerate if there is a vertex incident to more than d facets.)

A bottleneck in the method is the procedure used to find the edge set of the current polyhedron, which is executed at every iteration step. For this purpose, a necessary and sufficient adjacency condition (test) is usually verified for each pair of vertices. There are two well-known tests: algebraic and combinatorial. As a rule, the combinatorial test performs much faster than the algebraic one. Below, a new accelerated modification of the combinatorial test is proposed, which is referred to as a graph test. Another improvement is associated with a decrease in the number of pairs of vertices to be tested for adjacency. It was noted in [11] that some of the edges of the current polyhedron do not lead to the generation of new vertices at late iterations and it was shown how the required storage and the execution time can be considerably reduced on this basis. Below, this idea is further developed and used to modify the double description method with a dynamic order of adding inequalities. Theoretical results and a numerical experiment show that the modifications proposed are considerably superior in terms of speed to the original algorithm and other modifications, for example, in [11].

To be definite, consider the construction of the skeleton of a polyhedral cone defined by the homogeneous system of linear inequalities $Ax \ge 0$. Let the cone be acute. For an arbitrary cone, the problem can easily be reduced to the above one by passing from the original space $\mathbb{R}^d$ to the orthogonal complement of the subspace $\{x \in \mathbb{R}^d : Ax = 0\}$.

The necessary definitions and notation are introduced in Section 1. The scheme for the double description method is described in Section 2. In Section 3, we discuss several methods for adding inequalities to the original system. Sections 4 and 5 deal with two important procedures in the double description method. Specifically, in Section 4, we describe methods for verifying the adjacency of extreme rays in a polyhedral cone and propose a new graph modification of this test. In Section 5, techniques for reducing the number of generated pairs of adjacent extreme rays are presented and new solution methods for this problem are suggested. The modifications proposed have been implemented in the

SKELETON code, which is briefly outlined in Section 6. Numerical results are also presented in this section.

1. NOTATION AND PRELIMINARIES

The material in this section is based primarily on [7, 16]. The polyhedral cone in $\mathbb{R}^d$ (or simply a cone) is defined as a set

$$C = \{x \in \mathbb{R}^d : Ax \geq 0\},$$

where $A \in \mathbb{R}^{m \times d}$ is an $m \times d$ real matrix. The system of linear inequalities $Ax \geq 0$ is said to define the cone C. A cone is called *acute* if it does not contain nonzero subspaces. It is well known that a cone is acute if and only if $\operatorname{rank} A = d$, where $\operatorname{rank} A$ denotes the rank of the matrix A. Any polyhedral cone C can be defined as the conical hull of a finite set of vectors $u_1, u_2, \ldots, u_n$ in $\mathbb{R}^d$; i.e.,

$$C = \{x = \alpha_1 u_1 + \alpha_2 u_2 + \ldots + \alpha_n u_n : \alpha_i \geq 0, i = 1, 2, \ldots, n\}.$$

The set of vectors $u_1, u_2, \ldots, u_n$ are said to generate the cone C.

A nonzero vector $u \in C$ is said to be a *ray* of C. Two rays u and v are called *equal* (written as $u \simeq v$) if for some $\alpha > 0$ it is true that $u = \alpha v$. A ray $u \in C$ is said to be *extreme* if the condition $u = \alpha v + \beta w$ for $\alpha \geq 0$, $\beta \geq 0$, and $v, w \subseteq C$ implies $u \simeq v \simeq w$. The set of extreme rays of an acute cone is its minimal generating system and is called the skeleton of the cone. Suppose that P is a convex subset of $\mathbb{R}^d$ and, for some $a \in \mathbb{R}^d$ and $\alpha \in \mathbb{R}$, it holds that $P \subseteq \{x : ax \leq \alpha\}$. Then $P \cap \{x : ax = \alpha\}$ is called by a *face* of P. Two extreme rays u and v of an acute cone C are said to be *adjacent* if the minimal face containing both rays does not contain any other extreme rays of the cone. The skeleton of C is denoted by $U(C)$, and the set of all pairs $\{u, v\}$ of adjacent extreme rays is denoted by $E(C)$.

Given a matrix $A \in \mathbb{R}^{m \times d}$, $i \in \{1, 2, \ldots, m\}$, and $K \subseteq \{1, 2, \ldots, m\}$, let a_i denote the ith row of A and A_K denote the submatrix of A composed of the rows a_i, where $i \in K$. In expressions of the form ax, where $a \in \mathbb{R}^d$ and $x \in \mathbb{R}^d$, we interpret a as a row vector and x as a column vector.

2. DOUBLE DESCRIPTION METHOD

The main idea behind the double description method [3] for constructing the skeleton of a polyhedral cone is as follows. A matrix $A \in \mathbb{R}^{d \times n}$ with $\operatorname{rank} A = d$ is fed as input. At a preliminary stage, we find a subsystem $Bx \geq 0$ of the system $Ax \geq 0$ that consists of d inequalities of rank d. It is easy to see that the skeleton of the cone defined by this subsystem is formed of the columns of the matrix B^{-1}. Then the inequalities of the original system are added to the subsystem $Bx \geq 0$ one after another with the skeleton updated each time. The updating rules are given in the following theorem.

Theorem 1 (the main theorem of the double description method [3]). *Let $A \in \mathbb{R}^{m \times d}$, $\operatorname{rank} A = d$, and $a \in \mathbb{R}^d$. If U is the skeleton of the cone $C = \{x \in \mathbb{R}^d : Ax \geq 0\}$ and*

$$U_0 = \{u \in U : au = 0\}, \quad U_+ = \{u \in U : au > 0\}, \quad U_- = \{u \in U : au < 0\},$$

then the skeleton of the cone

$$C' = \{x \in \mathbb{R}^d : Ax \geq 0, ax \geq 0\}$$

is the union $U_+ \cup U_0 \cup U_\pm$, *where*

$$U_\pm = \{w = (au)v - (av)u : u \in U_+, v \in U_-, (u, v) \in E(C)\}.$$

The general scheme for the double description method is described below (see [3]). A matrix $A \in \mathbb{R}^{m \times d}$ with $\operatorname{rank} A = d$ is fed as input to the DDM algorithm. The output is the skeleton of the cone $\{x : Ax \geq 0\}$.

procedure DDM(A)
Find $K \subseteq \{1, 2, ..., m\}$ such that $|K| = d$, $\det A_K \neq 0$
Construct the skeleton U of the cone $\{x : A_K x \geq 0\}$
while $K \neq \{1, 2, ..., m\}$
Choose $i \in K$ (1)
$U_+ \leftarrow \{u \in U : a_i u > 0\}$
$U_- \leftarrow \{u \in U : a_i u < 0\}$
$U_0 \leftarrow \{u \in U : a_i u = 0\}$
$U_\pm \leftarrow \emptyset$
for each $u \in U_+$
for each $v \in U_-$
if u and v are adjacent in the cone $\{x : A_K x \geq 0\}$ (2)
$w \leftarrow (a_i u)v - (a_i v)u$
$U_\pm \leftarrow U_\pm \cup \{w\}$
end
end
end
$U \leftarrow U_+ \cup U_0 \cup U_\pm$
$K \leftarrow K \backslash \{i\}$
end
return U
end

A major feature of the double description method is that each of its iterations involves two complete descriptions of the current cone, namely, the system of inequalities $A_K x \geq 0$ and its skeleton, hence, the name.

Modifications of the double description method differ from each other, for example, in the following parameters:

(i) the order of considering the inequalities of the original system at step (1);

(ii) the method for determining the extreme rays of the skeleton at step (2);

(iii) the moment when the rays are tested for adjacency.

Numerous experiments show that the total running time of the algorithm depends substantially on the order in which the inequalities are considered (see, e.g., [2, 8, 11]). On the other hand, the procedure for constructing a set E of pairs of adjacent extreme rays takes much time at each iteration step.

Tests for verifying the adjacency of extreme rays at step (2) are described in Section 4.

Various authors have proposed modifications of the algorithm in which the set E is rebuilt as soon as the list of extreme vectors is updated (see, e.g., [8, 11]). Below is an example of such a modification.

procedure DDM.M1(A)
  Find $K \subseteq \{1, 2, \ldots, m\}$ such that $|K| = d$, $\det A_K \neq 0$
  Construct the skeleton U of the cone $\{x : A_K x \geq 0\}$
  $E \leftarrow \{\{u, v\} : u, v \in U, u \neq v\}$
  while $K \neq \{1, 2, \ldots, m\}$
    while $i \in K$ (1)'
    $U_+ \leftarrow \{u \in U : a_i u > 0\}$
    $U_- \leftarrow \{u \in U : a_i u < 0\}$
    $U_0 \leftarrow \{u \in U : a_i u = 0\}$
    $U_\pm \leftarrow \emptyset$
    $E_+ \leftarrow \{\{u, v\} \in E : u, v \in U_+ \cup U_0\}$ (3)
    $E_0 \leftarrow \emptyset$
    for each $\{u, v\} \in E$ (4)
      if $u \in U_+$ and $v \in U_-$
        $w \leftarrow (a_i u)v - (a_i v)u$
        $U_\pm \leftarrow U_\pm \cup \{w\}$
        $E_0 \leftarrow E_0 \cup \{\{u, w\}\}$
      else if $u \in U_-$ and $v \in U_+$
        $w \leftarrow (a_i v)u - (a_i u)v$
        $U_\pm \leftarrow U_\pm \cup \{w\}$
        $E_0 \leftarrow E_0 \cup \{\{v, w\}\}$
      end
    end
    $U \leftarrow U_+ \cup U_0 \cup U_\pm$
    Construct the set E'' of pairs of rays from U_0,
      adjacent in the cone $\{x \in \mathbb{R}^d : A_K x \geq 0, A_i x \geq 0$ (2)'
    $E \leftarrow E_+ \cup E' \cup E''$
    $K \leftarrow K \cup \{i\}$
  end
  return U
end

3. ORDER OF ADDING INEQUALITIES

Various methods for adding inequalities of the original system in the double description method have been proposed (see, e.g., [8, 11]). Each of them defines a rule for choosing the index k at step (1) and (1') of the algorithms DDM and DDM.M1, respectively.

In the methods minindex, lexmin, mincutoff, and minpairs, k is specified as

$$k_{\text{minindex}} = \min K, \quad k_{\text{lexmin}} = \arg\text{lexmin}\{a_i : i \in K\},$$

$$k_{\text{mincutoff}} = \arg\min\{|U_-| : i \in K\}, \quad k_{\text{minpairs}} = \arg\min\{|U_-| \cdot |U_+| : i \in K\}, \text{ respectively.}$$

For the methods maxindex, lexmax, maxcutoff, and maxpairs, min and lexmin in the formulas are replaced by max and lexmax, respectively. In the method random, the index k is selected from K at random with equal probabilities. Thus, at each iteration of the algorithm, mincutoff (maxcutoff) minimizes (respectively, maximizes) the number of cutoff extreme rays. The method minpairs (maxpairs) minimizes (respectively, maximizes) the number of considered potentially adjacent pairs.

The methods for adding inequalities can be divided into two groups:

(i) methods with a fixed order of inequalities;

(ii) methods with a dynamically determined order of inequalities.

In the case of (i), the order of selecting inequalities can be defined prior to the execution of iterations. These methods include minindex, maxindex, lexmin, lexmax, and random. In this case, the inequalities of the original system $Ax \geq 0$ can be sorted beforehand and $k = \min K$ can be assumed at step (1').

In the case of (ii), k cannot be known in advance. The methods with a dynamic order include mincutoff, maxcutoff, minpairs, and maxpairs.

Some results of an experimental comparison of these methods are presented in Section 6.

4. METHODS FOR VERIFYING THE ADJACENCY OF EXTREME RAYS

Below, we consider some well-known methods for verifying the adjacency of extreme rays of a cone and propose a new method. The necessary and sufficient conditions for the adjacency of rays given later can be used in the original algorithm DDM and its modification DDM.M1. Note that, at step (2') in DDM.M1, the set of extreme rays of the cone $\{x \in \mathbb{R}^d: A_K x \geq 0, a_k x \geq 0\}$ that belong to U_0 coincides with the set of all extreme rays of the cone $\{x \in \mathbb{R}^d : A_K x \geq 0, a_k x = 0\}$. This circumstance should be taken into account in the verification of the adjacency tests described below.

As usual, let $C = \{x : Ax \geq 0\}$, $A \in \mathbb{R}^{m \times d}$, $\operatorname{rank} A = d$, and $u \in \mathbb{R}^d$. Define $Z(u) = \{i : A_i u = 0\}$. Thus, $Z(u)$ is the index set of constraints of the original system $Ax \geq 0$ that are active for the vector u.

There are two well-known tests for the adjacency of extreme rays in a cone: algebraic and combinatorial.

Proposition 1 (algebraic test). *Let $u, v \in U(C)$. Then $\{u, v\} \in E(C)$ if and only if* $\operatorname{rank} A_{Z(u) \cap Z(v)} = d - 2$.

Proposition 2 (combinatorial test). *Let $u, v \in U(C)$. Then $\{u, v\} \in E(C)$ if and only if $Z(u) \cap Z(v) \subset Z(w)$ for any $w \in U(C) \backslash \{u, v\}$.*

The algebraic test is a consequence of the Minkowski theorem. The combinatorial test was first proposed in [3]. Its proof can be found in [4].

The rank in the algebraic test can be calculated using well-known linear algebra algorithms, which run in $Q(md^2)$ time. Thus, the complexity of constructing all pairs of adjacent rays by applying the algebraic test is $O(mn^2d^2)$.

Proposition 1 yields the following simple necessary condition for the adjacency of rays.

Proposition 3. *If $\{u, v\} \in E(C)$, then $Z(u) \cap Z(v) > r - 2$.*

This necessary condition has been considered by many authors (see, e.g., [4, 5, 8, 10, 11]). Numerous experiments suggest that it should be verified before the execution of any adjacency test.

Consider the combinatorial test in more detail. It states that extreme rays u and v of a cone C are adjacent if and only if the inequalities that are active for both rays are not both active for any other extreme ray; in other words, the minimal face containing both u and v does not contain any other extreme rays. Due to the last statement, the proof of the combinatorial test is obvious.

For the combinatorial test, it is convenient to define a matrix $T = (t_{ij}) \in \{0, 1\}^{n \times m}$, such that $t_{ij} = 1$ if and only if $a_j u_i > 0$, where $U = \{u_1, u_2, ..., u_n\}$. Rays u_i and $u_{i'}$ are adjacent if and only if, for any $k \in \{1,2, ..., n\} \backslash \{i, i'\}$, there is l such that

$$t_{il} = t_{i'l} = 0, \quad k_{kl} = 1.$$

The complexity of verifying the adjacency of two rays u and v is $O(mn)$. Thus, the complexity of constructing all pairs of adjacent rays by applying the combinatorial test is $O(mn^3)$.

We propose a new, graph modification of the combinatorial test that considerably speeds up the verification of the adjacency of extreme rays. Let G be a simple graph, i.e., an undirected graph without loops or multiple edges. It can be constructed from a cone C as follows. The vertex set of G is the set U of extreme rays of C, and $\{u, v\}$ forms an edge in G if and only if $|Z(n) \cap Z(v)| \geq r - 2$. The set of all edges of G is denoted by $E(G)$.

Proposition 4 (graph test). *Let $u, v \in U(C)$. Then $\{u, v\} \in E(C)$ if and only if there is no ray w in $U(C)$ other than u* or *v such that $\{u, w\} \in E(G)$, $\{v, w\} \in E(G)$, and $Z(u) \cap Z(v) \subseteq Z(w)$.*

Proof. Let $u, v \in U(C)$. According to Proposition 2, $\{u, v\} \in E(C)$ if and only if $Z(u) \cap Z(v) \subseteq Z(w)$ for no $w \in U(C)$. However, according to Proposition 3, this condition does not hold for any $w \in U(C)$ that is not adjacent in G to both u and v. Therefore, this condition is sufficient to be verified only for rays w such that $\{u, w\} \in E(G)$ and $\{v, w\} \in E(G)$.

Note that Proposition 4 can be used to verify the adjacency of extreme rays without constructing G. Instead, at each iteration, we can construct only a neighborhood D the current vertex u of this graph.

We obtain the algorithm Graph.Adj for finding all the pairs of adjacent extreme rays.

```
procedure Graph.Adj(U)
    E ← ∅
    S ← ∅
    for each u ∈ U
        D ← ∅
        S ← S ∪ {u}
        for each v ∈ U\{u}
            if |Z(u)\Z(v)| ≥ d − 2
                D ← D ∪ {v}
            end
        end
        for each v ∈ D\S
            if |Z(u)\Z(v)| ≥ d − 2
                if ∄ w ∈ D\{v}, Z(u)\Z(v) ⊆ Z(w)
                    E ← E ∪ {u, v}
                end
            end
        end
    end
    return E
end
```

The skeleton $U = U(C)$ of C is fed as input to the algorithm. For each extreme ray u, the set $Z(u)$ is assumed to be known. The algorithm returns the set E of all pairs of adjacent extreme rays.

The algorithm Graph.Adj also includes the verification of the necessary condition from Proposition 3.

Let δ denote the maximum of the vertex degrees in G. It is easy to see that the complexity of verifying the adjacency of two extreme rays u and v in Graph.Adj is $O(m\delta)$. The complexity of the entire algorithm Graph.Adj is $O(n(n + \delta^2 m))$. Since $\delta < n$, this complexity does not asymptotically exceed the upper bound $O(mn^3)$ for the complexity of solving the given problem by applying the combinatorial test. In many problems, $\delta \ll n$ and Graph.Adj becomes much more superior. The numerical results presented in Section 6 confirm this superiority.

5. DECREASE IN THE NUMBER OF CONSIDERED PAIRS OF ADJACENT RAYS

When a new inequality is added, the DDM algorithm enumerates all the pairs of extreme rays u and v such that $a_k u \cdot a_k v < 0$ and then test them for adjacency. In contrast, the DDM.M1 algorithm does enumerate all such pairs. Instead, the list E of only adjacent pairs of extreme rays is updated at each iteration. Note that the memory required for storing E can depend quadratically on the number of generated rays. In practice, this may lead to a shortage of memory space. On the other hand, there are pairs $\{u, v\} \in E$ that do not lead to the generation of a new ray. The detection of such pairs of rays at early stages reduces the required memory, since such pairs are eliminated from E. This also reduces the running time of step (3) and the number of iterations in loop (4). Moreover, the detection of such pair before it is tested for adjacency can save time needed for this verification.

First, we consider DDM.M1 with a fixed order of inequalities. Assume that the inequalities have been sorted in the necessary order. Therefore, $k = \min K$ can be assumed at step (1)' in DDM.M1. The following method, referred to as PlusPlus, was proposed in [11] for reducing the number of considered pairs of adjacent rays in DDM.M1.

Let $u, v \in E$ and $a_k u = a_k v = 0$. Compute $a_i u$, $a_i v$ ($i = 1, 2, ..., m$). Note that $a_i u \geq 0$, $a_i v \geq 0$ ($i = 1, 2, ..., k - 1$). Let, for some $k' > k$,

$$a_i u > 0, \quad a_i v \geq 0, \quad i = k + 1, 2, \ldots, k' - 1, \quad a_{k'} u < 0, \quad a_{k'} v < 0.$$

In this case, it is easy to see that $\{u, v\}$ does not lead to the generation of any new extreme ray at the subsequent iterations. Therefore, this pair is excluded from consideration; i.e., it is eliminated from E.

Table 1. Execution time of SKELETON for the problem ccc_7

The order of considering inequalities	Modif. PlusPlus	Modif. Graph. Adj	Time, s	Total number of constructed rays	Total number of constructed pairs of adjacent rays
(1)	(2)	(3)	(4)	(5)	(6)
lexmax	+	+	257	337803	443041
	+	−	372		
	−	+	519		14453733
	−	−	1302		
maxindex	+	+	3010	1087966	1313076
	+	−	6703		
	−	+	4582		41591800
	−	−	17169		
mincutoff	+	+	3615	1207557	10442749
	+	−	10764		
	−	+	4586		44135376
	−	−	17761		
lexmin	+	+	4285	1225951	1533651
	+	−	9458		
	−	+	7961		50631122
	−	−	23401		
maxcutoff	+	+	7566	1789770	8459167
	+	−	26128		
	−	+	9400		59093531
	−	−	41580		
minindex	+	+	8378	1691488	2125675
	+	−	19117		
	−	+	13480		73114625
	−	−	58506		
maxpairs	+	+	14719	2547932	12445583
	+	−	54832		
	−	+	16911		76720581
	−	−	82620		
minpairs	+	+	15468	2180085	12468623
	+	−	53931		
	−	+	19002		76176098
	−	−	94830		

We propose the following modification of PlusPlus for the DDM.M1 algorithm with a dynamic order of inequalities. Let $u, v \in E$ and $a_k u = a_k v = 0$. Compute $a_i u$, $a_i v$, $i = 1, 2, ..., m$. Note that $a_i u \geq 0$, $a_i v \geq 0$, $i = 1, 2, ..., k-1$. Let

$$a_i u \cdot a_i v \geq 0, \quad i = k+1, 2, \ldots, m.$$

In this case, it is easy to see that $\{u, v\}$ does not lead to the generation of any new extreme ray at the subsequent iterations. Therefore, this pair is excluded from consideration; i.e., it is eliminated from E.

Let us present numerical results demonstrating the superiority of the above modifications.

Table 2. Execution time of SKELETON for the problem ccp_7

The order of considering inequalities	Modif. PlusPlus	Modif. Graph. Adj	Time, s	Total number of constructed rays	Total number of constructed pairs of adjacent rays
(1)	(2)	(3)	(4)	(5)	(6)
lexmin	+	+	5502	1186741	1434203
	+	–	11932		
	–	+	9738		52433304
	–	–	37635		
mincutoff	+	+	9413	1921944	14673772
	+	–	26500		
	–	+	12300		73419448
	–	–	50989		
minindex	+	+	9589	1815547	2264908
	+	–	21153		
	–	+	16475		80150892
	–	–	66086		
maxindex	+	+	37293	3712906	4384170
	+	–	90336		
	–	+	53458		150592933
	–	–	259914		
lexmax	+	+	42412	4072635	4774940
	+	–	105143		
	–	+	58899		158090265
	–	–	287400		
minpairs	+	+	60920	4131333	20266953
	+	–	211875		
	–	+	76248		152416312
	–	–	397073		
maxcutoff	+	+	67566	5501375	28885642
	+	–	259762		
	–	+	78645		180817307
	–	–	422795		
maxpairs	+	+	78205	5935122	30214584
	+	–	302448		
	–	+	95036		196189215
	–	–	507222		

6. NUMERICAL EXPERIMENT

The code SKELETON (http://www.uic.unn.ru/~zny/skeleton) has been developed by the author. It involves all the above modifications of the double description method. The code supports limited-accuracy integer arithmetic (with data represented as 4-byte integers), arbitrary accuracy integer arithmetic (with integers of unlimited size), and double-precision floating-point arithmetic. S.V. Lobanov has made to the code available through the Internet (http://www.arageli.org/skeletondemo).

The experiments were performed on an Intel Core 2 CPU 6300 1.86 GHz, 2 Gb RAM, Microsoft Windows XP Professional, Version 2002, SP2 computer system with the use of the C++ MS Visual Studio 2005 compiler with the option /î2 switched off.

Table 3. Comparison of the performances of SKELETON and cdd

Problem	Order	Input		Output	Execution time, s	
		m	*d*	*n*	Skeleton	cdd
cube16	lexmin	17	32	65536	10	27
cube18	minindex	19	36	262144	190	1103
mit729-9	lexmin	8	729	4862	97	57
ccc7	lexmax	63	21	38780	271	4232
ccp7	lexmin	64	22	116765	5681	15981

Tables 1 and 2 present the results of a numerical experiment with the input specified as the skeleton of a complete cut cone ccc7 (63 vectors in $\mathbb{R}^{21}$) and the vertices of a complete cut polyhedron ccc_7 (64 points in $\mathbb{R}^{21}$) [17]. The code computed their facet descriptions (38780 and 116764 inequalities, respectively) in limited-accuracy integer arithmetic. The first column indicates the order of considering inequalities of the system. The second and third columns say whether PlusPlus and Graph.Adj were used. The fourth column gives the time (in seconds) required for the code to solve the problem. The last two columns provide the total number of constructed extreme rays (over all iterations of the algorithm) and the total number of generated pairs of adjacent rays, respectively.

The tables show that, as a rule, the code performs considerably better with the use of Graph.Adj and PlusPlus.

Table 3 compares the execution times of SKELETON (with Graph.Adj and PlusPlus switched on) and the cdd code by K. Fukuda (http://www.ifor.math.ethz.ch/~fukuda/cdd_home/cdd.html). The latter is based on the DDM.Ml algorithm with certain modifications from [11]. The experiment was performed for the problems described in [11] in double-precision floating-point arithmetic.

A parallel version of the algorithm and the corresponding code can be found in [18].

ACKNOWLEDGMENTS

The author is grateful to V.N. Shevchenko for helpful discussions and to A.A. Bader, S.S. Lyalin, and S.V. Lobanov for valuable advice concerning the code. This work was supported by the Russian Foundation for Basic Research, project no. 09-01-00545-a.

REFERENCES

1. F. Preparata and M. Shamos, *Computational Geometry: An Introduction* (Springer-Verlag, New York, 1985; Mir, Moscow, 1989).
2. D. Avis, D. Bremner, and R. Seidel, "How Good Are Convex Hull Algorithms?" Comput. Geom. Theory Appl. **7** (5, 6), 265–301 (1997).
3. T. S. Motzkin, H. Raiffa, G. L. Thompson, and R. M. Thrall, "The Double Description Method," *Contributions to the Theory of Games* (Princeton University Press, Princeton, N. J., 1953; Fizmatgiz, Moscow, 1961).
4. E. Burger, "Uber homoge nelineare Ungleichungssysteme," Angewandte Math. Mech. **36** (3/4), 135–139 (1956).
5. N. V. Chernikova, "Algorithm for Finding a General Formula for the Nonnegative Solutions of a System of Linear Equations," USSR Math. Math. Phys. **4** (4), 151–158 (1964).
6. N. V. Chernikova, "Algorithm for Finding a General Formula for the Nonnegative Solutions of a System of Linear Inequalities," USSR Math. Math. Phys. **5** (2), 228–233 (1965).
7. S. N. Chernikov, *Linear Inequalities* (Nauka, Moscow, 1968) [in Russian].
8. S. I. Veselov, I. E. Parubochii, and V. N. Shevchenko, "A Software Program for Finding the Skeleton of the Cone of Nonnegative Solutions of a System of Linear Inequalities," in *System and Applied Software Programs* (Gor'kov. Gos. Univ., Gor'kii, 1984), pp. 83–92 [in Russian].
9. F. Fernandez and P. Quinton, "Extension of Chernikova's Algorithm for Solving General Mixed Linear Programming Problems," Res. Rep. RR-0943 (INRIA, Rennes, 1988).
10. H. Le Verge, "A Note on Chernikova's Algorithm," Res. Rep. RR-1662 (INRIA, Rennes, 1992).
11. K. Fukuda and A. Prodon, "Double Description Method Revisited," *Combinatorics and Computer Science* (Springer-Verlag, New York, 1996), pp. 91–111.

12. V. N. Shevchenko and A. Yu. Chirkov, "On the Complexity of Finding the Skeleton of a Cone," *Proceedings of 10th All-Russia Conference on Mathematical Programming and Applications* (Ural. Otd. Ross. Akad. Nauk, Yekaterinburg, 1997), p. 237.
13. V. N. Shevchenko and D. V. Gruzdev, "A Modification of the Fourier–Motzkin Algorithm for Constructing Triangulations," Diskret. Anal. Issled. Oper., Ser. 2. **10** (10), 53–64 (2003).
14. O. L. Chernykh, "Construction of the Convex Hull of a Finite Set of Points Using Triangulation," USSR Comput. Math. Math. Phys. **31** (8), 80–86 (1991).
15. B. Chaselle, "An Optimal Convex Hull Algorithm in Any Fixed Dimension," Discrete Comput. Geom., No. 10, 377–409 (1993).
16. A. Schrijver, *Theory of Linear and Integer Programming* (Wiley, Chichester, 1986; Mir, Moscow, 1991).
17. M. M. Deza and M. Laurent, *Geometry of Cuts and Metrics* (Springer-Verlag, Berlin, 1997; MTsNMO, Moscow, 2001) [in Russian].
18. N. Yu. Zolotykh and S. S. Lyalin, "A Parallel Algorithm for Finding the General Solution of the System of Linear Inequalities," Vestn. Nizhegorod. Gos. Univ., No. 5, 193–199 (2009).