SKELETON 02.00.04 Manual

Nikolai Yu. Zolotykh*

October 6, 2009

Abstract

This paper describes SKELETON: implementation of several new variations of well-known Double Description Method (DDM) for solving the vertex and facet enumeration problems for convex polyhedra. New enhancements makes SKELETON quite competitive in comparison with other implementations of DDM. The source code of SKELETON 02.00.04 is available at `http://www.uic.nnov.ru/~zny/skeleton`.

Contents

*N.I. Lobachevsky State University of Nizhni Novgorod, Russia

1 What's new?

Skeleton 02.00.00 May 7, 2006
It is new, completely re-written, fast version of Skeleton.

Skeleton 02.00.01 November 1, 2006
Skeleton now runs on Linux platform. Source code is available.

Skeleton 02.00.02 November 7, 2006
Floating point arithmetic is now supported.

Skeleton 02.00.03 May 30, 2007
Time bug fixed. Vertices of 3d polyhedra are in clockwise or anticlockwise direction.

Skeleton 02.00.04 October 6, 2009
A bug occuring on 64 bit architecture fixed. (Thanks to Sergey Lyalin and Sergey Lobanov.)

2 Introduction

It is well known that any polyhedon in $\mathbf{R}^d$ can be represented by the following two ways:

(1) as a set of solutions to the system of linear inequalities, or

(2) as the (Minkowski's) sum of the conic hull of some vectors and the convex hull of some points in $\mathbf{R}^d$.

The problem to generate representation (2) if representation (1) is available is called the *vertex enumeration problem*. The converse one is called the *facet enumeration problem*, or *convex hull problem*.

Analogously, any polyhedral cone in $\mathbf{R}^d$ can be represented by the following two ways:

(1) as a set of solutions to the system of homogenius linear inequalities, or

(2) as a set of all non-negative linear combinations of some vectors in $\mathbf{R}^d$.

There is a standard way to reduce vertex/facet enumeration problem for polyhedra to the correspondent problem for polyhedral cones. From theoretical point of view it is convenient to consider both problems just for polyhedral cones.

The program Skeleton implements several variations of Double Description Method (DDM) [MRTT53] solving the vertex and facet enumeration problems. DDM is considered in a few papers and monographs [Bur56, Che64, Che65, Che68a, VPS84, Che68b, FQ88, Ver92, FP96, SC97, SG03].

Skeleton works with the system of linear inequalities whose entries are integers (arbitrary precision or 4 bytes long ints) or reals (double floating point numbers).

In our implementation we use ideas descibed in [VPS84, FP96, SC97] and some new enhancements. All these makes SKELETON quite competitive in comparison with other implementations of DDM, in particular, [Ver92, Fuk02, Gru03]. Early version of SKELETON is described in [Zol97].

SKELETON can be distributed under the terms of GNU GENERAL PUBLIC LICENSE Version 2. Read file COPYING.

Thanks to Sergey Lobanov you can use Skeleton on-line (withput install it). Visit `http://www.arageli.org`.

3 Theoretical Preliminaries

3.1 Polyhedral Cones

Polyhedral cone C is the set of all solutions to a system of homogenius linear inequalities and equations $Ax \geq 0$, $Bx = 0$:

$$C = \left\{ x \in \mathbf{R}^d : \ Ax \geq 0, \ Bx = 0 \right\}, \tag{$\star$}$$

where $A \in \mathbf{R}^{m \times d}$, $B \in \mathbf{R}^{t \times d}$ (m or/and t may be equal to 0 that corresponds to the case when inequalities or/and equations are absent accordingly). The case of system of equations and inequalities can be obviously reduced to the case of system with only inequalities. For this instead of $Bx = 0$ we can consider $Bx \geq 0$ and $-Bx \geq 0$. But it will be more convinient to consider the more general case.

The maximal subspace contained in the cone C can be described as a set of all solutions to the system $Ax = 0$, $Bx = 0$. The dimension of this subspace is equal to $d - \operatorname{rank}(A^\top, B^\top)$. The cone is called *pointed* if it contains only zero subspace, that's equivalent to $\operatorname{rank}(A^\top, B^\top) = d$.

Let $a \in \mathbf{R}^d$, $a \neq 0$. The hyper-plane $\left\{ x \in \mathbf{R}^d : \ ax = 0 \right\}$ is called *supporting* for the cone C if $C \subseteq \{x : \ ax \geq 0\}$ or $C \subseteq \{x : \ ax \leq 0\}$. The intersection of the cone with a supporting hyper-plain is called its *face*.

Consider the system of vectors $u_1, \dots, u_p$ in $\mathbf{R}^d$. The *linear hull* $\operatorname{Lin}(u_1, \dots, u_p)$ of the system is the set of all linear combinations of these vectors:

$$\operatorname{Lin}(u_1, \dots, u_p) = \{\lambda_1 u_1 + \ldots + \lambda_p u_p : \ \lambda_i \in \mathbf{R} \ (i = 1, \dots, p)\}.$$

The *non-negative*, or *conic*, *hull* $\operatorname{NonNeg}(u_1, \dots, u_p)$ of the system is the set of all non-negative linear combinations:

$$\operatorname{NonNeg}(u_1, \dots, u_p) = \{\lambda_1 u_1 + \ldots + \lambda_p u_p : \ \lambda_i \in \mathbf{R}, \ \lambda_i \geq 0 \ (i = 1, \dots, p)\}.$$

Theorem 1 (Minkowski) *For any polyhedral cone C in $\mathbf{R}^d$ there exist vectors $u_1, \dots, u_p$, $v_1, \dots, v_q$ in $\mathbf{R}^d$ such that*

$$C = \operatorname{Lin}(u_1, \dots, u_p) + \operatorname{NonNeg}(v_1, \dots, v_q). \tag{$\star\star$}$$

Obviously, w.l.g. in Minkowski's theorem we can omit $\mathrm{Lin}(u_1,\ldots,u_p)$ item and instead of ($\star\star$) write simply $C = \mathrm{NonNeg}(v_1,\ldots,v_q)$.

Using matrix notation we can re-formulate Minkowski's theorem as follows. For any matrices $A \in \mathbf{R}^{m\times d}$ and $B \in \mathbf{R}^{t\times d}$ there exist matrices $V \in \mathbf{R}^{q\times d}$ and $U \in \mathbf{R}^{p\times d}$ such that

$$\left\{x \in \mathbf{R}^d : \; Ax \geq 0, \; Bx = 0\right\} = \{x = \mu V + \lambda U : \; \mu \in \mathbf{R}^n, \; \mu \geq 0, \; \lambda \in \mathbf{R}^s\}.$$

Vectors $u_1,\ldots,u_p$, $v_1,\ldots,v_q$ (equvalently, matrices U and V) can be chosen in such a way that the following *properties of minimality* hold:

1. $p = d - \mathrm{rank}(A^\top, B^\top)$ and $\mathrm{Lin}(u_1,\ldots,u_p)$ is the maximal subspace L included in C (so, the system $u_1,\ldots,u_p$ is a basis of L); and

2. q is minimal among all possible q such that ($\star\star$) holds (this means also that the system $v_1,\ldots,v_q$ is irreducible); in this case the system $v_1,\ldots,v_q$ is called a *skeleton* of the cone C.

The vectors in a skeleton are unique up to any positive multiplyer and any item in L. If the cone C is pointed, i.e. $\mathrm{rank}(A^\top, B^\top) = d$ and, hence, $p = 0$, then $v_1,\ldots,v_q$ (and — up to positive multiplyer — only they) are *extreme rays* of C. We'll say that two vectors in a skeleton are *adjacent* if minimal face containing both does not contain any other vector in the skeleton.

The converse theorem to Minkowski's one is correct and is known as Weyl's theorem.

Theorem 2 (Weyl) *For any vectors $u_1,\ldots,u_p$, $v_1,\ldots,v_q$ in $\mathbf{R}^d$ there exist matrices $A \in \mathbf{R}^{m\times d}$ and $B \in \mathbf{R}^{t\times d}$ such that*

$$\mathrm{Lin}(u_1,\ldots,u_p) + \mathrm{NonNeg}(v_1,\ldots,v_q) = \left\{x \in \mathbf{R}^d : \; Ax \geq 0, \; Bx = 0\right\}.$$

Obviously, w.l.g. in Weyl's theorem we can omit linear equations $Bx = 0$ and simply write $\mathrm{Lin}(u_1,\ldots,u_p) + \mathrm{NonNeg}(v_1,\ldots,v_q) = \left\{x \in \mathbf{R}^d : \; Ax \geq 0\right\}$.

Using matrix notation we can re-formulate Weyl's theorem as follows. For any matrices $U \in \mathbf{R}^{p\times d}$ and $V \in \mathbf{R}^{q\times d}$ there exist matrices $A \in \mathbf{R}^{m\times d}$ and $B \in \mathbf{R}^{t\times d}$ such that

$$\{x = \lambda U + \mu V : \; \lambda \in \mathbf{R}^p, \; \mu \in \mathbf{R}^q, \; \mu \geq 0\} = \left\{x \in \mathbf{R}^d : \; Ax \geq 0, \; Bx = 0\right\}.$$

Matrices A and B can be chosen in such a way that the following *properties of minimality* hold:

1. $t = d - \mathrm{rank}\,\{u_1,\ldots,u_p,v_1,\ldots,v_q\}$ and $\{x : Bx = 0\}$ is the minimal subspace containing C (this means also that the system $Bx = 0$ is irreducible); and

2. m is minimal among all possible m such that ($\star$) holds (this means also that the system $Ax \geq 0$ is irreducible).

The rows in such a matrix A are unique up to any positive multiplyer and any item which is linear combinations of rows in B. The rows in A correspond to faces of maximum dimension. In particular, if the cone C is full-dimensional, i.e. $\operatorname{rank}\{u_1, \ldots, u_p, v_1, \ldots, v_q\} = d$ and, hence, $t = 0$, then the rows in A correspond to *facets* of C.

These theorems suggest two fundamental problems. First one is to obtain a dual representation $(\star\star)$ if a representation $(\star)$ is known. The second problem is converse. It turns out that these problems are computationaly equivalent as the following theorem shows. So, we can concentate on the first problem.

Theorem 3 (Farkas–Minkowski–Weyl) *If*

$$C = \left\{x \in \mathbf{R}^d : \ Ax \geq 0, \ Bx = 0\right\} = \{x = \mu V + \lambda U : \ \mu \in \mathbf{R}^q, \ \mu \geq 0, \ \lambda \in \mathbf{R}^p\}$$

then

$$C' = \left\{x \in \mathbf{R}^d : \ Vx \geq 0, \ Ux = 0\right\} = \left\{x = \mu A + \lambda B : \ \mu \in \mathbf{R}^m, \ \mu \geq 0, \ \lambda \in \mathbf{R}^t\right\}.$$

Moreover, if rows in V and U form a skeleton of C and a basis of minimal subspace correspondingly, then $Vx \geq 0$, $Ux = 0$ are irreducible systems determining C' and viceversa. Analogous property is true for A and B.

3.2 Polyhedra

Polyhedron P is the set of all solutions to a system of linear inequalities and equations $Ax \geq b$, $Bx = c$:

$$P = \left\{x \in \mathbf{R}^d : \ Ax \geq b, \ Bx = c\right\}, \tag{$*$}$$

where $A \in \mathbf{R}^{m \times d}$, $B \in \mathbf{R}^{t \times d}$, $b \in \mathbf{R}^m$, $c \in \mathbf{R}^t$ (m or/and t may be equal to 0 that corresponds to the case when inequalities or/and equations are absent accordingly). The case of system of equations and inequalities can be obviously reduced to the case of system with only inequalities. But it will be more convinient to consider the more general case.

Let $a \in \mathbf{R}^d$, $a \neq 0$, $\alpha \in \mathbf{R}$. The hyper-plane $\left\{x \in \mathbf{R}^d : \ ax = \alpha\right\}$ is called *supporting* for the polyhedron P if $P \cap \{x : \ ax = \alpha\} \neq \emptyset$ and $P \subseteq \{x : \ ax \geq \alpha\}$ or $P \subseteq \{x : \ ax \leq \alpha\}$. The intersection of the polyhedron with a supporting hyper-plane is called a *face* of the polyhedron. The face whith dimension 0 (i.e. a point) is called a *vertex* of P.

Consider the system of vectors $w_1, \ldots, w_s$ in $\mathbf{R}^d$. The *convex hull* $\operatorname{Conv}(w_1, \ldots, w_s)$ of the system is the set of all *convex combinations* of these vectors, i.e.:

$$\operatorname{Conv}(w_1, \ldots, w_s) = \left\{\lambda_1 w_1 + \ldots + \lambda_s w_s : \ \lambda_i \in \mathbf{R}, \ \lambda_i \geq 0, \ \sum_{i=1}^{s} \lambda_i = 1\right\}.$$

The set of points in $\mathbf{R}^d$ which can be represented as a convex hull of some finite system of points is called *polytope*.

From Minkowski's theorem we get the following.

Theorem 4 *For any polyhedron P in $\mathbf{R}^d$ there exist vectors $u_1, \ldots, u_p$, $v_1, \ldots, v_q$, $w_1, \ldots, w_s$ in $\mathbf{R}^d$ such that*

$$P = \mathrm{Lin}(u_1, \ldots, u_p) + \mathrm{NonNeg}(v_1, \ldots, v_q) + \mathrm{Conv}(w_1, \ldots, w_s). \qquad (**)$$

So, any polyhedron is the sum of a cone and a polytope.

Obviously, w.l.g. in the theorem we can omit $\mathrm{Lin}(u_1, \ldots, u_p)$ item and instead of $(**)$ write simply $P = \mathrm{NonNeg}(v_1, \ldots, v_q) + \mathrm{Conv}(w_1, \ldots, w_s)$.

Vectors $u_1, \ldots, u_p$, $v_1, \ldots, v_q$, $w_1, \ldots, w_s$ can be chosen in such a way that the following *properties of minimality* hold:

1. $u_1, \ldots, u_p$ is a basis of the subspace L associated with the maximal linear variety in P; and
2. q and s are minimal among all possible q and s such that $(**)$ holds (this means also that the systems $v_1, \ldots, v_q$ and $w_1, \ldots, w_s$ are irreducible).

In this case vectors $w_1, \ldots, w_s$ are unique up to any item in L; vectors $v_1, \ldots, v_q$ are unique up to any positive multiplyer and any item in L. If $p = 0$ then points $w_1, \ldots, w_s$ (and only they) are vertices of P and vectors $v_1, \ldots, v_q$ (and only they) are extreme rays of P.

The problem of constructing the representation $(**)$ if representation $(*)$ is available is called the *vertex enumeration problem.* It can be reduced to the analogous problem for cones as follows.

Consider the cone in $\mathbf{R}^{d+1}$

$$C = \left\{(x_1, \ldots, x_n, x_{n+1})^\top \in \mathbf{R}^{d+1} : \ Ax \geq bx_{n+1}, \ Bx = cx_{n+1}, \ x_{n+1} \geq 0\right\},$$

where $x = (x_1, \ldots, x_n)^\top$. For the cone C we can get a dual representation

$$C = \mathrm{Lin}(\overline{u}_1, \ldots, \overline{u}_p) + \mathrm{NonNeg}(\overline{v}_1, \ldots, \overline{v}_q)$$

for some vectors $\overline{u}_1, \ldots, \overline{u}_p$, $\overline{v}_1, \ldots, \overline{v}_q$ in $\mathbf{R}^{d+1}$.

Let $\overline{u}_i = (u_i, u_{i,n+1})$ $(i = 1, \ldots, p)$, $\overline{v}_i = (v_i, v_{i,n+1})$ $(i = 1, \ldots, q)$. Since the system of homogenius linear inequalities and equations contains the inequality $x_{n+1} \geq 0$, then it is clear that $u_{i,n+1} = 0$ $(i = 1, \ldots, p)$. Suppose w.l.g. that $v_{i,n+1} = 0$ $(i = 1, \ldots, s)$, $v_{i,n+1} \neq 0$ $(i = s+1, \ldots, q)$. Now it is not hard to see that the initial polyhedron P has the following dual representation:

$$\begin{aligned} P = {} & \mathrm{Lin}(u_1, \ldots, u_p) + \\ & + \mathrm{NonNeg}(v_1, \ldots, v_s) + \mathrm{Conv}\left(\frac{1}{v_{s+1,d+1}} \cdot v_{s+1}, \ldots, \frac{1}{v_{q,d+1}} \cdot v_q\right). \end{aligned}$$

Moreover, if the system $\overline{u}_1, \ldots, \overline{u}_p$ is a basis of the maximal linear subspace in C and the system $\overline{v}_1, \ldots, \overline{v}_q$ is a skeleton of C then the system of vectors constructed to describe P also has the property of minimality. In particular, if $p = 0$ then $\dfrac{1}{v_{s+1,d+1}} \cdot v_{s+1}, \ldots, \dfrac{1}{v_{q,d+1}} \cdot v_q$ are vertices of P.

From Weyl's theorem we get the following.

Theorem 5 *For any vectors $u_1, \ldots, u_p$, $v_1, \ldots, v_q$, $w_1, \ldots, w_s$, in $\mathbf{R}^d$ there exist matrices $A \in \mathbf{R}^{m\times d}$ and $B \in \mathbf{R}^{t\times d}$ and vectors $b \in \mathbf{R}_m$, $c \in \mathbf{R}^t$ such that*

$$\begin{aligned}\mathrm{Lin}(u_1, \ldots, u_p) + \mathrm{NonNeg}(v_1, \ldots, v_q) + \mathrm{Conv}(w_1, \ldots, w_s) = \\ = \left\{x \in \mathbf{R}^d : \ Ax \geq b, \ Bx = c\right\}.\end{aligned}$$

Matrices A and B can be chosen in such a way that the following *properties of minimality* hold:

1. the system $Bx = c$ is irreducible and $\{x : Bx = c\}$ is the minimal linear variety containing P; and
2. m is minimal among all possible m such that $(*)$ holds (this means also that the system $Ax \geq 0$ is irreducible).

In this case the rows in the matrix (A, b) correspond to faces of maximum dimension. In particular, if P is full-dimensional, i.e.

$$\mathrm{rank}\,\{u_1, \ldots, u_p, v_1, \ldots, v_q, w_1 - w_s, \ldots, w_{s-1} - w_s\} = d$$

and, hence, $t = 0$, then the rows in (A, b) correspond to *facets* of P.

The problem of constructing the representation $(*)$ if representation $(**)$ is available is called the *facet enumeration problem*, or the *convex hull problem.* It can be reduced to the analogous problem for cones as follows.

In $\mathbf{R}^{d+1}$ consider the cone

$$C = \mathrm{Lin}(\overline{u}_1, \ldots, \overline{u}_p) + \mathrm{NonNeg}(\overline{v}_1, \ldots, \overline{v}_q, \overline{w}_1, \ldots, \overline{w}_s),$$

where $\overline{u}_i = (u_i, 0)$ $(i = 1, \ldots, p)$, $\overline{v}_i = (v_i, 0)$ $(i = 1, \ldots, q)$, $\overline{w}_i = (w_i, 1)$ $(i = 1, \ldots, s)$ and find its representation

$$C = \left\{(x_1, \ldots, x_d, x_{d+1})^\top \in \mathbf{R}^{d+1} : \ Ax - bx_{d+1} \geq 0, \ Bx - cx_{d+1} = 0\right\},$$

where $x = (x_1, \ldots, x_d) \in \mathbf{R}^d$. Now it is not hard to see that the initial polyhedron P has the following representation:

$$P = \left\{x \in \mathbf{R}^d : \ Ax \geq b, \ Bx = c\right\}.$$

Moreover, if A, B, b, c are such that each of the systems $Ax - bx_{d+1} \geq 0$ and $Bx = cx_{d+1}$ is irreducible and $\left\{(x_1, \ldots, x_d, x_{d+1})^\top \in \mathbf{R}^{d+1} : \ Bx - cx_{d+1} = 0\right\}$ is the minimal subspace containing C then the system of inequalities and equations constructed to describe P also has a property of minimality.

3.3 The main idea of the algorithm

Given a matrix $A \in \mathbf{R}^{m\times d}$, DDM generates a basis of maximal subspace and a skeleton of the cone $C = \left\{x \in \mathbf{R}^d : \ Ax \geq 0\right\}$. Obviously, the case then the cone is defined by a system of linear inequalities and equations can be reduced to the case with only inequalities.

In the preliminary step of DDM the rank r of A and a basis of the maximal subspace containing in C are founded. Also, a skeleton of the cone determined by some irreducible subsystem containing r inequalities is generated. Then, other inequalities are added one after the other and every time the skeleton is re-conctructed. Consider this slightly in detail.

Let K be a cone determined by some subsystem of $Ax \geq 0$. Suppose that a skeleton of K is known. Consider what will happen with the skeleton then a new inequality $ax \geq 0$ is added.

Each vector in the skeleton of K falls to one of the following sets:

1. W_0 is the set of all vectors w in the skeleton such that $aw = 0$;
2. W_+ is the set of all vectors w in the skeleton such that $aw \geq 0$;
3. W_- is the set of all vectors w in the skeleton such that $aw \leq 0$.

A skeleton of the new cone is formed by all elements in W_+ and W_0 and vectors which we obtain as follows. For each pair of vectors $w' \in W_+$ and $w'' \in W_-$ adjacent in K we obtain their linear combination w satisfying to equality $aw = 0$. Every such w should be included to the skeleton of the new cone.

Variations of DDM differs one from another by ordering in which inequalities are choose from the system, the methods used to find adjacent rays, a time when the adjacency is computed and others [VPS84, FP96, SC97]. Checking the adjacency seems the most time-expensive procedure in DDM and different techniques to determine what pairs of vectors should be verifying are used [FP96].

4 How to Install

The source code of Skeleton is available at `http://uic.nnov.ru/~zny/skeleton`. The package contains a documentation and two C++ files: `skeleton.cpp` and `ddm.hpp`.

Skeleton uses Arageli library [Ara09]. To compile Skeleton first of all you should install Arageli. Suppose that you have Arageli installed on your computer.

To compile the code in standard Linux environment you need `gcc` version 4. Type

```
g++ -O2 skeleton.cpp -o skeleton -larageli
```

Files `skeleton.cpp` and `ddm.hpp` are supposed to be in the current directory. Also, `gcc` must know the locations of Arageli include files and library.

To compile the code in Windows you can use MS Visual Studio C++ Compiler version 6.0 or later. Type

```
cl -O2 skeleton.cpp arageli.lib
```

5 How to Use

Given a matrix $A \in \mathbf{Z}^{m\times d}$, program Skeleton generates a basis of maximal subspace and a skeleton of the cone $C = \{x \in \mathbf{R}^d : \ Ax \geq 0\}$.

To use Skeleton, first of all, one should prepare file with your data. This file must contain the size and entries of matrix A. Entries of A should be integer. Numbers are separated by spaces and blank lines. For example, if you want to find a skeleton of the cone C defined as a set of solution to the system

$$\begin{cases} x_1 & & & \geq 0, \\ -x_1 & & +x_3 + x_4 & \geq 0, \\ & -x_2 & +x_3 & \geq 0, \\ & & x_3 + x_4 & \geq 0, \\ x_1 & +x_2 & +x_4 & \geq 0, \\ -x_1 & -x_2 & -x_4 & \geq 0 \end{cases}$$

then the input file (say `ex00.ine`) is

To run Skeleton just type in the command prompt:

```
skeleton filename
```

where `filename` is the name of the input file. Example:

```
skeleton ex00.ine
```

Skeleton produces two files: "output" file, "log" file and "summary" file. By default, their names are obtained by adding extention `.out`, `.log`, `.sum` respectively to input file name. In our example Skeleton produces files `ex00.ine.out`, `ex00.ine.log`, and `ex00.ine.sum`.

The output file contains sizes and entries of matrix U (vectors of a basis in row-wise order) and matrix V (vectors of a skeleton). Also, the file can contain other information (it depends on options used; see the list of available options below). In our example we get the following file `ex00.ine.out`: Thus, we get $u_1 = (0, -1, -1, 1)^\top$, $v_1 = (1, -1, 1, 0)^\top$, $v_2 = (0, 0, 1, 0)^\top$ and $C = \mathrm{Lin}(u_1) + \mathrm{NonNeg}(v_1, v_2)$.

The log file contains computation hystory. By default, this information is also displayed on stdcrt during computation. In our example we get the following file `ex00.ine.log`:

The summary file contains computation summary. By default, this information are also displayed on stdcrt after computation. In our example we get the following file `ex00.ine.sum`:

You may set different options affecting the process of computation and the output of information:

```
skeleton filename options
```

where `options` is a list of options. Each option is an abbreviation beginning with minus sign. Options are separated by spaces. Example:

```
skeleton ex00.ine -lexmin -adjacency
```

Complete list of available options is in the next section.

6 Options

The following options are available (the default parameters are in braces):

{`-minindex`}, `-lexmin`, `-lexmax`, `-random`, `-mincutoff`, `-maxcutoff`, `-minedges`, `-maxedges`
: These options affect the ordering of inequalities to be added at each iteration of DDM.

`-prefixedorder`, `-noprefixedorder`
: If options `-mincutoff -maxcutoff -minedges -maxedges` are chosen then only `-noprefixedorder` is possible. In other cases both options are available; the default one is `-prefixedorder`.

{`-graphinc`}, `-nographinc`
: These affect the way of determining adjacent vectors.

{`-plusplus`}, `-noplusplus`
: If option `-plusplus` is chosen then only the pairs of adjacent vectors that will be necessary on the future iterations are constructed. If option `-noplusplus` is chosen then all edges are constructed on each iteration.

{`-bigint`}, `-int`, `-float`
: By default, arbitrary precision integer arithmetic is used. Option `-int` forces to use ordinary (4 bytes) integer precision arithmetic. Option `-float` forces to use double floating point (8 bytes) arithmetic.

`-zerotol value`
: The option affects only if option `-float` is used. This is used to change a zero tolerance for floating point computation. A real value is considered as zero if its absolute value is at most the tolerance. The default value for the zero tolerance is `1e-8`.

{`-adjacency`}, `-noadjacency`
: Option `-adjacency` forces to find also all pairs of adjacent vectors in the skeleton.

`-inputfile filename`
: This option defines input file name. `skeleton -inputfile filename` is equivalent to `skeleton -inputfile filename`. By default, all input is from stdin.

`-outputfile filename`
: This option sets the name of output file. By default, this name is obtained by adding extention `.out` to input file name. If input was from stdin then output file is `skeleton.out`.

`-logfile filename`
: This option sets the name of log file. By default, this name is obtained by adding extention `.log` to input file name. If input was from stdin then log file is `skeleton.log`.

`-summaryfile filename`
: This option sets the name of summary file. By default, this name is

obtained by adding extention `.sum` to input file name. If input was from stdin then log file is `skeleton.sum`.

{`-outputinfile`}, `-nooutputinfile`
: If `-nooutputinfile` is chosen then SKELETON will not put results in output file.

`-outputonstdout`, {`-nooutputonstdout`}
: If `-outputonstdin` is chosen then SKELETON will put results on stdout.

{`-loginfile`}, `-nologinfile`
: If `-nologinfile` is chosen then SKELETON will not put log information in log file.

{`-logonstdout`}, `-nologonstdout`
: If `-nologonstdout` is chosen then SKELETON will not put log information on stdout.

{`-summaryinfile`}, `-nosummaryinfile`
: If `-nosummaryinfile` is chosen then SKELETON will not put summary information in summary file.

{`-summaryonstdout`}, `-nosummaryonstdout`
: If `-nosummaryonstdout` is chosen then SKELETON will not put summary information on stdout.

`-ine`, {`-noine`}
: If `-ine` is chosen then SKELETON will put the input matrix A on stdout and in outputfile. This works only if option `-outputinfile` or `-outputonstdout` correspondingly turns on.

{`-bas`}, `-nobas`
: If `-nobas` is chosen then SKELETON will not put the matrix U (with entries of basis of maximal subspace contained in the cone) on stdout and in outputfile. This works only if option `-outputinfile` or `-outputonstdout` correspondingly turns on.

{`-ext`}, `-noext`
: If `-noext` is chosen then SKELETON will not put the matrix V (with entries of skeleton vectors) on stdout and in outputfile. This works only if option `-outputinfile` or `-outputonstdout` correspondingly turns on.

`-inc`, {`-noinc`}
: If `-inc` is chosen then SKELETON will put the matrix $VA^\top$ on stdout and in outputfile. This works only if option `-outputinfile` or `-outputonstdout` correspondingly turns on.

`-incext`, {`-noincext`}
: If `-incext` is chosen then for each vector in skeleton the program will print (on stdout and in outputfile) iniqualities which hold as equality. This works only if option `-outputinfile` or `-outputonstdout` correspondingly turns on.

`-incine`, {`-noincine`}
: If `-incine` is chosen then for each inequality in the initial system the

program will print (on stdout and in outputfile) vectors in the skeleton for which the inequality holds as equality. This works only if option `-outputinfile` or `-outputonstdout` correspondingly turns on.

`-matrices`, `-nomatrices`
: `-matrices` is equivalent to `-ine`, `-ext`, `-bas`, `-inc`; `-nomatrices` is equivalent to `-noine`, `-noext`, `-nobas`, `-noinc`. This works only if option `-outputinfile` or `-outputonstdout` turns on.

{`-summary`}, `-nosummary`
: If `-nosummary` is chosen no summary information (input/output/log file names, sizes of matrices and option values) will not put on stdout and in summary file. This works only if option `-summaryinfile` or `-summaryonstdout` turns on.

{`-time`}, `-notime`
: If `-notime` is chosen no time information (the time of beginning and ending of computation and complete time elapsed) will not put on stdout and in summary file. This works only if option `-summaryinfile` or `-summaryonstdout` turns on.

`-help` or `--help`
: `skeleton -help` prints the list of available options and terminates the program. `skeleton --help` does the same.

`-version` or `--version`
: `skeleton -version` prints SKELETON version and terminates the program. `skeleton --version` does the same.

7 More Examples

7.1 Cube With a Cutted Vertex

Consider the polyhedron described by the following system:

$$\begin{cases} x_1 & & & \geq 0, \\ & x_2 & & \geq 0, \\ & & x_3 & \geq 0, \\ x_1 & & & \leq 1, \\ & x_2 & & \leq 1, \\ & & x_3 & \leq 1, \\ 2x_1 & + 2x_2 & + 2x_3 & \leq 5. \end{cases} \tag{1}$$

It is a cube with a "cutted" vertex (see Fig. 1).

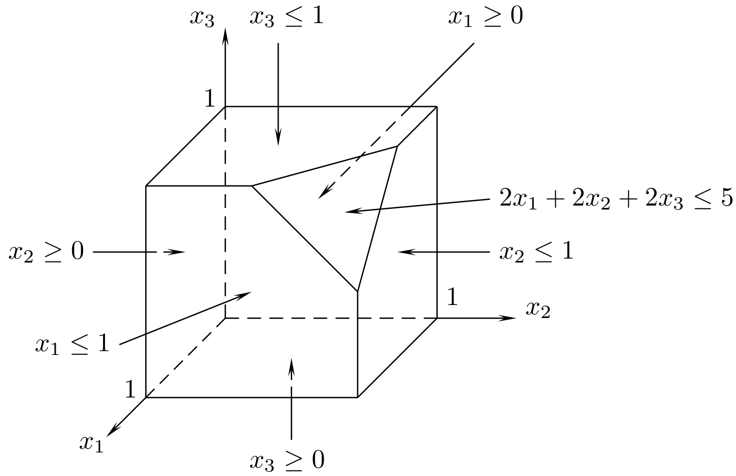

Figure 1: Cube with a cutted vertex. Facet representation

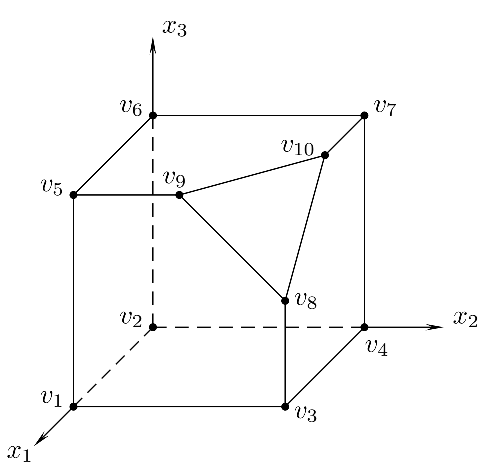

Figure 2: Cube with a cutted vertex. Vertex representation

The corresponding cone is described by the following homogenious system:

$$\left\{\begin{array}{rrrrl} x_1 & & & & \geq 0, \\ & x_2 & & & \geq 0, \\ & & x_3 & & \geq 0, \\ -x_1 & & & + \ x_4 & \geq 0, \\ & - \ x_2 & & + \ x_4 & \geq 0, \\ & & - \ x_3 & + \ x_4 & \geq 0, \\ -2x_1 & - \ 2x_2 & - \ 2x_3 & + \ 5x_4 & \geq 0, \\ & & & x_4 & \geq 0 \end{array}\right. \tag{2}$$

(setting $x_4 = 1$ we get the initial system). So, input file (named `cwcv.ine`) is

```
8 4
  0  1  0  0
  0  0  1  0
  0  0  0  1
  1 -1  0  0
  1  0 -1  0
  1  0  0 -1
  5 -2 -2 -2
  1  0  0  0
```

Running Skeleton with

```
skeleton cwcv.ine -adjacency -incine
```

we get the following file `cwcv.ine.out`:

```
* Basis:
0 4
* Extreme rays:
10 4
1 1 0 0
1 1 1 0
2 2 2 1
1 1 0 1
2 2 1 2
1 0 0 1
1 0 1 1
2 1 2 2
1 0 1 0
1 0 0 0
```

Matrix with entries of the basis is empty (it contains 0 rows), hence the polyhedron does not contain any non-zero linear variety. Matrix with entries of the skeleton has 10 rows, hence the polyhedron has 10 vertex (see Fig. 2). The forth coordinate corresponds to the denominator in entries of all these vertices. They are $v_1 = (1, 0, 0)^\top$, $v_2 = (0, 0, 0)^\top$, $v_3 = (1, 1, 0)^\top$, $v_4 = (0, 1, 0)^\top$,

$v_5 = (1,0,1)^\top$, $v_6 = (0,0,1)^\top$, $v_7 = (0,1,1)^\top$, $v_8 = \left(1,1,\frac{1}{2}\right)^\top$, $v_9 = \left(1,\frac{1}{2},1\right)^\top$, $v_{10} = \left(\frac{1}{2},1,1\right)^\top$.

Also, we have computed all pairs of adjacent vertices. The polyhedron has 15 edges. They are v_1–v_2, v_1–v_3, v_1–v_5, v_2–v_4, v_2–v_6, v_3–v_4, v_3–v_8, v_4–v_7, v_5–v_6, v_5–v_9, v_6–v_7, v_7–v_{10}, v_8–v_9, v_8–v_{10}, v_9–v_{10}.

Information concerning "Inequalities Incidence" tell us that 7 facets are formed by vertices v_2, v_4, v_6, v_7; v_1, v_2, v_5, v_6; v_1, v_2, v_3, v_4; v_1, v_3, v_5, v_8, v_9; v_3, v_4, v_7, v_8, v_{10}; v_5, v_6, v_7, v_9, v_{10}; v_8, v_9, v_{10} correspondingly.

Now we can check our computations by "reversing" them. Form the input file (named `cwcv.ext`) containing entries of vertices found: and evoke SKELETON:

```
skeleton cwcv.ext
```

We get the following file `cwcv.ext.out`: Since the matrix with "basis" is empty the polyhedron has full dimension. The matrix with "skeleton" has 7 rows. They correspond to exactly the same inequalities as in (1), so the polyhedron has 7 facets. We remark that in the list obtained there is no row corresponding to the inequality $x_4 \geq 0$ because in our case it is redundant in (2).

7.2 Voronoi Diagram

Let W be a system of s points in $\mathbf{R}^d$. For each $w \in W$ we can consider the set

$$\mathrm{VS}(w) = \left\{x \in \mathbf{R}^d : \ \forall v \in W \setminus \{w\} \ \ \mathrm{dist}(x,p) \leq \mathrm{dist}(x,q)\right\},$$

where dist is the Euclidean distance function. The set $\mathrm{VS}(w)$ is called a *Voronoi cell*. It is a polyhedron. Its vertices are called *Voronoi vertices* and extreme rays are called *Voronoi rays*. The set $\{\mathrm{VS}(w) : \ w \in W\}$ of all Voronoi cells is called *Voronoi diagram* (see Fig. 3). For generating Voronoi diagram the following construction is widely used.

For each $w \in W$ consider the hyperplane tangent at $w = (w_1, \ldots, w_d)^\top$ to the paraboloid $\left\{(x_1, \ldots, x_d, x_{d+1}) : \ x_{d+1} = x_1^2 + \ldots + x_d^2\right\}$. This hyperplane is represented by the following equation:

$$-2w_1x_1 - \ldots - 2w_dx_d + x_{d+1} + w_1^2 + \ldots + w_d^2 = 0.$$

Replacing the equality with inequality $\geq$ and considering these inequalities for each $w \in W$ we get the system of s linear inequalities. Let P be the polyhedron of all solutions to the system. It turns out that P is a lifting of Voronoi diagram to one higher dimensional space; and the projection of each facet of P associated with w is exactly the Voronoi cell $\mathrm{VS}(w)$. The vertices and extreme rays of P project exactly to the Voronoi vertices and rays, respectively [Fuk04].

As an example consider the set of points $(0,0)^\top$, $(2,0)^\top$, $(-2,0)^\top$, $(0,1)^\top$, $(1,2)^\top$, $(-1,2)^\top$, $(0,3)^\top$. For generating their Voronoi diagram consider the

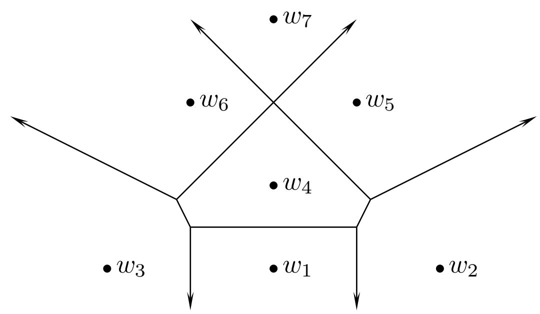

Figure 3: Voronoi diagram for the set of points

system

$$\begin{cases} \qquad\qquad\quad + x_3 \qquad\quad \geq 0, \\ -4x_1 \qquad\quad + x_3 + 4x_4 \geq 0, \\ \;\;4x_1 \qquad\quad + x_3 + 4x_4 \geq 0, \\ \qquad\; - 2x_2 + x_3 + \;\; x_4 \geq 0, \\ -2x_1 - 4x_2 + x_3 + 5x_4 \geq 0, \\ \;\;2x_1 - 4x_2 + x_3 + 5x_4 \geq 0, \\ \qquad\; - 6x_2 + x_3 + 9x_4 \geq 0, \\ \qquad\qquad\qquad\qquad\quad x_4 \geq 0. \end{cases}$$

Prepare file `exvoronoi.ine`: Now evoke SKELETON:

```
skeleton exvoronoi.ine -incine -incext
```

We get the following file `exvoronoi.ine.out`:

Each row in "skeleton" with last entry equal to 0 corresponds to a Voronoi ray. Each row whose last entry is non-zero corresponds to a Voronoi vertex. So, we get 5 vertices of Voronoi diagram (dividing by the forth component and ignoring the third one):

$$v_1 = \left(-1,\ \frac{1}{2}\right)^\top, \quad v_2 = \left(1,\ \frac{1}{2}\right)^\top, \quad v_3 = \left(\frac{7}{6},\ \frac{5}{6}\right)^\top,$$

$$v_4 = \left(-\frac{7}{6},\ \frac{5}{6}\right)^\top, \quad v_5 = (0,\ 2)^\top,$$

and 5 extreme rays (ignoring the third component):

$$v_6 = (0,-1)^\top,\ v_7 = (2,1)^\top,\ v_8 = (-2,1)^\top,\ v_9 = (1,1)^\top,\ v_{10} = (-1,1)^\top.$$

Interpret "Inequalities Incidence" we get Fig. 4.

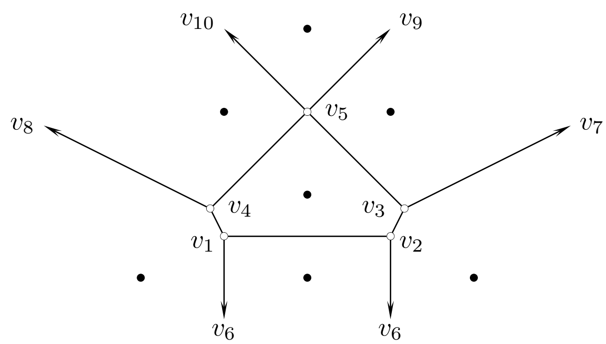

Figure 4: Voronoi diagram constructed with the help of SKELETON

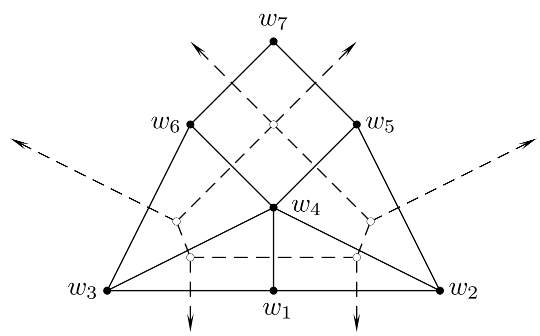

Figure 5: Delaunay triangulation is dual to Voronoi diagram

7.3 Delaunay Triangulation

Let W be a system of s points in $\mathbf{R}^d$ and v be some Voronoi vertex for W. The convex hull of the nearest neighbor set of v is called the *Delaunay cell* of v. The *Delaunay complex* (or *triangulation*) of W is a partition of $\operatorname{Conv} W$ into the Delaunay cells of Voronoi vertices.

The Delaunay complex is not in general a triangulation but becomes a triangulation when the points in W are in *general position* (or *nondegenerate*), i.e. no $d+2$ points are cospherical or equivalently there is no point $c \in \mathbf{R}^d$ whose nearest neighbor set has more than $d+1$ elements.

The Delaunay complex is dual to the Voronoi diagram in the sense that there is a natural bijection between the two complexes which reverses the face inclusions (see Fig. 5) [Fuk04].

So, to generate Delaunay triangulation we can perform the following procedure. For each vertex of polyhedra (we are not interesting in extreme rays) in previous section we determine all facets incident to the vertex. Interpreting information about "Skeleton Incidence" in `exvoronoi.ine.out` we get that Delaunay cells in this example are formed by the following vertices w_1, w_3, w_4; w_1, w_2, w_4; w_2, w_4, w_5; w_3, w_4, w_6; w_4, w_5, w_6, w_7 (see Fig. 5).

There is a direct way to construct the Delaunay triangulation. Consider the same paraboloid as in the previous section: $x_{d+1} = x_1^2 + \ldots + x_d^2$. For each point $w = (w_1, \ldots, w_d)^\top$ in W consider its lifting $(w_1, \ldots, w_d, w_1^2 + \ldots + w_d^2)^\top$ in $\mathbf{R}^{d+1}$ and take the convex hull P of all such lifted points. Let $v = (0, \ldots, 0, 1)$. It turns out that any facet of $P + \mathrm{NonNeg}(v)$ which is not parallel to v is a Delaunay cell once its last coordinate is ignored, and any Delaunay cell is represented this way [Fuk04].

For our example form the file `exdelaunay.ext`: and evoke Skeleton:

```
skeleton exdelaunay.ext -incext
```

We get the file `exdelaunay.ext.out`: Only first 5 facets are not parallel to v (because their last coordinates are non-zero). So, we again have 5 Delaunay cells which are formed by points w_1, w_3, w_4; w_1, w_2, w_4; w_2, w_4, w_5; w_3, w_4, w_6; w_4, w_5, w_6, w_7 correspondingly (see Fig. 5).

References

[Ara09] Arageli: a library for doing exact computation. http://www.arageli.org, 2006-2009.

[Bur56] E. Burger. Über homogene lineare ungleichungssysteme. *Zeitschrift für Angewandte Mathematik und Mechanik*, 36:135–139, 1956.

[Che64] N.V. Chernikova. Algorithm for finding a general formula for the non-negative solutions of system of linear equations. *U.S.S.R. Computational Mathematics and Mathematical Physics*, 4(4):151–158, 1964.

[Che65] N.V. Chernikova. Algorithm for finding a general formula for the non-negative solutions of system of linear inequalities. *U.S.S.R. Computational Mathematics and Mathematical Physics*, 5(2):228–233, 1965.

[Che68a] S.N. Chernikov. *Linear inequalities*. Nauka, Moscow, 1968. Russian.

[Che68b] N.V. Chernikova. Algorithm for discovering the set of all solutions of a linear programming problem. *U.S.S.R. Computational Mathematics and Mathematical Physics*, 8(6):282–293, 1968.

[FP96] K. Fukuda and A. Prodon. Double description method revisited. In M. Deza, R. Euler, and I. Manoussakis, editors, *Lecture*

Notes in Computer Science, volume 1120, pages 91–111. Springer-Verlag, 1996. ps file available from ftp.ifor.math.ethz.ch, directory /pub/fukuda/reports.

[FQ88] F. Fernández and P. Quinton. Extension of Chernikova's algorithm for solving general mixed linear programming problems. Technical report, IRISA, Rennes, France, 1988.

[Fuk02] K. Fukuda. cdd, cddplus and cddlib homepage. http://www.cs.mcgill.ca/ fukuda/software/cdd home/cdd.html, 2002.

[Fuk04] K. Fukuda. Frequently asked questions in polyhedral computation. http://www.ifor.math.ethz.ch/staff/fukuda/polyfaq/polyfaq.html, 2004.

[Gru03] D.V. Gruzdev. Experimental comparison of algorithms for constructing convex hulls and triangulations. In O.B. Lupanov, editor, *Proceeding of the XIV International Workshop "Synthesys and Complexity of Control Systems*, pages 24–26, Nizhni Novgorod, 2003. Nizhni Novgorod Pedagogical University. Russian.

[MRTT53] T.S. Motzkin, H. Raiffa, G.L. Thompson, and R.M. Thrall. The double description method. In H.W. Kuhn and A.W.Tucker, editors, *Contributions to Theory of Games*, volume 2, Princeton, RI, 1953. Princeton University Press.

[SC97] V.N. Shevchenko and A.Yu. Chirkov. On complexity of constructing the skeleton of the cone. In *X Russian conference "Mathematical programming and applications"*, page 237, Ekaterinburgh, 1997. Ural department of Russian Academy of Science. Russian.

[SG03] V.N. Shevchenko and D.V. Gruzdev. Modification of Fourie–Motzkin algorithm for constructing triangulations. *Discrete Analisys and Operations Research, Series 2*, 10(10):53–64, 2003. Russian.

[Ver92] H.Le. Verge. A note on Chernikova's algorithm. Technical Report 635, IRISA, Campus de Beaulieu, Rennes, France, 1992.

[VPS84] S.I. Veselov, I.E. Parubochiĭ, and V.N. Shevchenko. A program for finding the skeleton of the cone of nonnegative solutions of a system of linear inequalities. In *Systems and Applied Programs. Part 2*, pages 83–92, Gorky, 1984. Gorky State University. Russian.

[Zol97] N.Yu. Zolotykh. Program implementation of Motzkin–Bürger algorithm for finding the skeleton of a polyhedral cone and its applications. In M.A. Antonets, V.E. Alekseyev, and V.N. Shevchenko, editors, *Proceeding of the 2nd International Conference "Mathematical Algorithms"*, pages 72–74, Nizhni Novgorod, 1997. Nizhni Novgorod State University. Russian.