cdd/cdd+ Reference Manual

Komei Fukuda
Institute for Operations Research
ETH-Zentrum, CH-8092 Zurich, Switzerland
fukuda@ifor.math.ethz.ch
and
Department of Mathematics
ETFL, CH-1015 Lausanne, Switzerland

(cdd ver. 0.61 and cdd+ ver. 0.76, March 17, 1999)

1 What's new?

Thanks to TU-Berlin's Polymake Team, cdd+ now runs with GNU's GMP rational library as well as with the GNU g++ Rational library. In particular, Polymake Team (Ewgenij Gawrilow and Michael Joswig) wrote C++ wrappers "gmp_integer.cc" and "gmp_rational.cc" for GMP which cdd+ can now use instead of GNU's G++ Rational arithmetic library.

Since cdd+ with GMP runs substantially faster than cddr+ with g++Rational, any cdd+ user who needs exact arithmetic computation is strongly recommended to use the GMP version. One should be also aware that GMP-2.0.2 has known bugs and all the patches available from GMP-Homepage (http://www.matematik.su.se/ tege/gmp/) should be applied first. Even with all these troubles, its speed compensates them easily. How much faster? It all depends on your data. For kkd* polytopes included in cdd+, cddr+_gmp runs three to ten times faster than cddr+_g++. For less complicated polytopes, the speedup might be modest but one can expect the gmp version runs always faster.

2 Introduction

The program cdd+ (cdd, respectively) is a C++ (ANSI C) implementation of the Double Description Method [MRTT53] for generating all vertices (i.e. extreme points) and extreme rays of a general convex polyhedron given by a system of linear inequalities:

$$P = \{x \in R^d : Ax \leq b\}$$

where A is an $m \times d$ real matrix and b is a real m dimensional vector. See, [FP96] for an efficient implementation of the double description method which is employed in cdd+.

One useful feature of cdd/cdd+ is its capability of handling the dual (reverse) problem without any transformation of data. The dual problem is known to be the *(convex) hull problem* which is to obtain a linear inequality representation of a convex polyhedron given as the Minkowski sum of the convex hull of a finite set of points and the nonnegative hull of a finite set of points in R^d: $P = conv(v_1, \ldots, v_n) + nonneg(r_1, \ldots, r_s)$, where the *Minkowski sum of two subsets S and T of R^d* is defined as $S + T = \{s + t \mid s \in S \text{ and } t \in T\}$. As we see in this manual, the computation can be done in straightforward manner. There is one assumption for the input for hull computation: the polyhedron must be full-dimensional.

Besides these basic functions, cdd/cdd+ can solve the general linear programming (LP) problem to maximize (or minimize) a linear function over polyhedron P. It is useful mainly for solving dense LP's with large m (say, up to few hundred thousands) and small d (say, up to 100).

The program cdd+ is a C++ program, converted from the ANSI C program cdd in 1995. Both programs have been updated for a few times since then. One major advantage of this C++-version over the C version is that it can be compiled for both rational (exact) arithmetic and floating point arithmetic. Note that cdd runs on floating arithmetic only. Since cdd+ uses GNU g++ library, in particular Rational library, one needs a recent (2.6.3 or higher) gcc compiler and g++-lib. One should be also warned that the computation can be considerably (10 - 100 times or even more) slower if the rational arithmetic is used. My idea is to keep the ANSI C code cdd as simple as as possible, while the C++ code cdd+ will be used to be an experimental platform to test new ideas.

The program cdd/cdd+ reads input and writes output in *Polyhedra format* which was defined by David Avis and the author in 1993, and has been updated in 1997. The program called lrs [Avi97] developed by David Avis is a C-implementation of the reverse search algorithm [AF92] for the same enumeration purpose, and it conforms to Polyhedra format as well. Hopefully, this compatibility of the two programs enables users to use both programs for the same input files and to choose whichever is useful for their purposes. From our experiences with relatively large problems, the two methods are both useful and perhaps complementary to each other. In general, the program cdd+ tends to be efficient for highly degenerate inputs and the program rs tends to be efficient for nondegenerate or slightly degenerate problems.

Among the hardest problems that could be solved (in floating-point arithmetic) by cdd+ is a 21-dimensional hull problem given by 64 vertices. This polytope, known as the *complete cut polytope on 7 points*, has exactly 116,764 facets and some of facets contain many vertices. It took 205 hours (eight and half days!) for cdd to compute the facets exactly on a SUN SparkServer 1000. The input file (ccp7.ine) of this polytope is included in the distribution. A considerably easier problem is ccc7.ine which is a variation of the problem (see e.g. [Gri90]).

The size of an input file hardly indicates the degree of hardness of its vertex/ray enumeration. While this program can handle a highly degenerate problem (prodmT5.ine) with 711 inequalities in 19 dimension quite easily with the computation time 1-2 minutes on a fast workstation, a 8-dimensional problem (mit729-9.ine) with 729 inequalities can be extremely hard. It takes two days to compute all (only 4862) vertices by a SUN SparkServer 1000. The latter problem arises from the ground state analysis of a ternary alloy model, see [CGAF94]. Both input files are included in the distribution.

Although the program can be used for nondegenerate inputs, it might not be very efficient. For nondegenerate inputs, other available programs, such as the reverse search code lrs or qhull (developed by the Geometry Center), might be more efficient. See Section 11 for pointers to these codes. The paper [ABS97] contains many interesting results on polyhedral computation and experimental results on cdd+, lrs, qhull and porta.

This program can be distributed freely under the GNU GENERAL PUBLIC LICENSE. Please read the file COPYING carefully before using.

I will not take any responsibility of any problems you might have with this program. But I will be glad to receive bug reports or suggestions at the e-mail addresses above. Finally, if cdd+ turns out to be useful, please kindly inform me of what purposes cdd has been used for. I will be happy to include a list of applications in future distribution if I receive enough replies. The most powerful support for free software development is user's appreciation and collaboration.

3 Polyhedra H- and V-Formats (Version 1997)

Every convex polyhedron has two representations, one as the intersection of finite halfspaces and the other as Minkowski sum of the convex hull of finite points and the nonnegative hull of finite directions. These are called H-representation and V-representation, respectively.

Naturally there are two basic Polyhedra formats, H-format for H-representation and V-format for V-representation. These two formats are designed to be almost indistinguishable, and in fact, one can almost pretend one for the other. There is some asymmetry arising from the asymmetry of two representations.

First we start with the halfspace representation. Let A be an $m \times d$ matrix, and let b be a column m-vector. The Polyhedra format (*H-format*) of the system $Ax \le b$ of m inequalities in d variables $x = (x_1, x_2, \ldots, x_d)^T$ is

```
various comments
H-representation
begin
m    d+1    numbertype
 b     -A
end
various options
```

where numbertype can be one of integer, rational or real. When rational type is selected, each component of b and A can be specified by the usual integer expression or by the rational expression "p/q" or "$-p/q$" where p and q are arbitrary long positive integers (see the example input file rational.ine). In the new 1997 format, we introduced "H-representation" which must appear before "begin". There was one restriction in the old polyhedra format (before 1997): the last d rows must determine a vertex of P. This is obsolete now.

Now we introduce Polyhedra *V-format.* Let P be represented by n extreme points and s rays as $P = conv(v_1, \ldots, v_n) + nonneg(r_1, \ldots, r_s)$. Then the Polyhedra V-format for P is defined as

```
various comments
V-representation
begin
n+s    d+1    numbertype
  1      v_1
  ⋮       ⋮
  1      v_n
  0      r_1
  ⋮       ⋮
  0      r_s
end
various options
```

Here we do not require that vertices and rays are listed separately; they can appear mixed in arbitrary order. Before the year 1997, the option "hull" was used instead of "V-representation" in V-format. The old option is obsolete but cdd+ still understands this option for backward compatibility. The reverse search code lrs has employed this new format from version 3.2.

When the representation statement, either "H-representation" or "V-representation", is omitted, the former "H-representation" is assumed.

It is strongly suggested to use the following rule for naming H-format files and V-format files:

(a) use the filename extension ".ine" for H-files (where ine stands for inequalities), and

(b) use the filename extension ".ext" for V-files (where ext stands for extreme points/rays).

The program cdd+ does two transformations, one from an H-format to a V-format, and the reverse. While an input file (in H-format or V-format) can have redundant information, cdd+ outputs a minimal representation (in V-format or H-format).

For example, let P be the following unbounded 3-dimensional H-polyhedron given by

$$P = \{x \in R^3 : 1 \leq x_1 \leq 2,\ 1 \leq x_2 \leq 2,\ 1 \leq x_3,\ x_1 + x_2 \leq 4\},$$

which is a 3-cube without one "lid". The last inequality is redundant because it is implied by $x_1 \leq 2$ and $x_2 \leq 2$. This is added to show how cdd+ works with redundant data. For finding all vertices and extreme rays, the input file for cdd+ is

```
* file name: ucube.ine
* 3 cube without one "lid"
H-representation
begin
    6      4    integer
  2   -1   0   0
  2    0  -1   0
 -1    1   0   0
 -1    0   1   0
 -1    0   0   1
  4   -1  -1   0
end
incidence
adjacency
input_adjacency
input_incidence
```

The meaning of options "incidence", "adjacency" "input_adjacency" and "input_incidence" will be explained in Section 4. After you run cddr+ (the rational arithmetic version of cdd+) or cddf+ with this input file, you will get an output file ucube.ext which is the minimal V-representation of the polyhedron:

```
* cdd+: Double Description Method in C++:Version 0.76 (March 17, 1999)
* Copyright (C) 1999, Komei Fukuda, fukuda@ifor.math.ethz.ch
* Compiled for Rational Exact Arithmetic with GMP
*Input File:ine/ucube.ine(6x4)
*HyperplaneOrder: LexMin
*Degeneracy preknowledge for computation: None (possible degeneracy)
*Vertex/Ray enumeration is chosen.
*Output adjacency file is requested.
*Input adjacency file is requested.
```

```
*Output incidence file is requested
*Input incidence file is requested.
*Computation completed at Iteration 6.
*Computation starts     at Wed Mar 17 06:56:51 1999
*            terminates at Wed Mar 17 06:56:51 1999
*Total processor time = 0 seconds
*                     = 0h 0m 0s
*FINAL RESULT:
*Number of Vertices =4, Rays =1
V-representation
begin
5  4  rational
 1 2 1 1
 1 1 1 1
 1 1 2 1
 1 2 2 1
 0 0 0 1
end
hull
```

The output shows that the polyhedron has four vertices $(2, 1, 1)$, $(1, 1, 1)$, $(1, 2, 1)$, $(2, 2, 1)$ and only one extreme ray $(0, 0, 1)$. The comments contain information on the name of input file, and the options chosen to run the program which will be explained in the next section.

Now, if you run cdd+ with this output file ucube.ext, cdd+ will perform the convex hull operation to recover essentially the original inequalities. More precisely, if we make a copy ucube2.ext of ucube.ext, and run cddr+ (or cddf+) with this copy, it will create a new ucube2.ine file:

```
* cdd+: Double Description Method in C++:Version 0.76 (March 17, 1999)
* Copyright (C) 1999, Komei Fukuda, fukuda@ifor.math.ethz.ch
* Compiled for Rational Exact Arithmetic with GMP
*Input File:ucube2.ext(5x4)
.
.
.
*Number of Facets = 5
H-representation
begin
5  4  rational
 -1 0 1 0
 -1 1 0 0
 2 0 -1 0
 2 -1 0 0
 -1 0 0 1
end
```

It is easy to check that this H-representation is essentially the same as the one we started with, except the new one does not contain any redundant inequality and the orderings are different.

Note that this back-and-forth transformation of a polyhedron works only when a polyhedron admits a unique minimal V-representation and a unique minimal H-representation. For example, when a polyhedron is full dimensional and contains at least one vertex, it satisfies these conditions.

4 Options

The following options are available for cdd. These options are set if they appear in input file after the "end" command. Independent options can be set simultaneously, but each option must be written separately in one line, and two options should not be written in one line. When two or more non-independent options are specified, the last one overrides the others. Also note that options are case-sensitive.

hull option

This old option exists only for the sake of backward compatibility. This is to enforce the computation to be convex hull computation, interpreting the current data as a V-representation. Use V-representation instead.

verify_input option

When this option is chosen, the program will output the input problem as cdd+ interpreted. The default output file is "*.solved". This option helps user to verify what problem is actually solved. The default for this option is off. See the sample files verifyinput1.ine and verifyinput2.ine

dynout_off option

When this option is chosen, the program will not output vertices and rays to the CRT in real time. The default is dynout_on.

stdout_off option

When this option is chosen, the program will not output any progress report of computation (iteration number. etc). The default is stdout_on.

logfile_on option

When this option is chosen, the program will output to a specified file (*.ddl) some information on the computation history. This can be useful when the user does not know which hyperplane order (mincutoff, maxcutoff, mixcutoff, lexmin, lexmax, minindex, random) is efficient for computation.

incidence, input_incidence, #incidence options

When the **incidence** option is selected, the incidence relation for each **output** with respect to input will be generated. The default filename is *.icd if **output** is inequalities (i.e. *.ine), and *.ecd [1] if **output** is extreme points and rays (i.e. *.ext).

When the **input_incidence** option is selected, the incidence relation for each **input** with respect to output will be generated. The default filename is *.icd if **input** is inequalities (i.e. *.ine), and *.ecd if **input** is extreme points and rays (i.e. *.ext). This option was added to cdd+ ver. 0.74 and **not available in cdd**.

Here, an extreme point is said to be *incident with* an inequality if the inequality is satisfied by equality. An extreme ray r is said to be *incident with* an inequality $a^T\ x \leq b$ if $a^T\ r = 0$.

[1] In earlier versions, *.icd was used for the new *.ecd file.

For example, since the incidence option was set for the example input file ucube.ine in the previous section, the program outputs the following ucube.ecd file:

```
*Incidences of output(=vertices/rays) and input (=hyperplanes)
*   for each output, #incidence and the set of hyperplanes containing it
*   or its complement with its cardinality with minus sign
*cdd input file : ucube.ine  (6 x 4)
*cdd output file: ucube.ext
begin
  5  6  7
 3 :  1 4 5
 3 :  3 4 5
 3 :  2 3 5
 -3 :  3 4 7
 -1 :  5
end
```

After "begin", there are three numbers 5   6   7. The first number 5 is a number of output (vertices and rays). The next number 6 is m, the number of inequalities in the input file. The last number 7 is usually $m+1$, and m if the input linear inequality system is homogeneous (i.e., has zero RHS) or the hull option is chosen. The number $m+1$ corresponds to the infinity constraint which is added for vertex/ray enumeration when the input system is not homogeneous.

The incidence data starts right after these three numbers. At each line, the cardinality of incident inequalities and the list of their indices are given. There is an exception that, when there are more incident inequalities than non-incident ones, then the program outputs the list of non-incident inequalities with its size with negative sign. This is to save space of output.

For example, the first output line 3 : 1 4 5 corresponds to the first vertex of ucube.ext file in previous section, that is, the vertex $(2, 1, 1)$. The first number 3 is simply the number of incident inequalities and the rest is the indices of those inequalities, and so the 1st, 4th and 5th inequalities are satisfied by equality at this vertex. The last output -1 : 5 corresponds to the ray $(0, 0, 1)$. Since all inequalities except the last (5th) inequality are incident with this ray, the output is the (shorter) complementary list with its cardinality (=1) with negative sign. Note that the full list would be 6 : 1 2 3 4 6 7, where 6 is the infinity plane. One can ignore the infinity plane for some purposes, but for analyzing the combinatorial structure of polyhedra, it is very important information.

Also, since the input_incidence option was set for the example input file ucube.ine in the previous section, the program outputs the following ucube.icd file:

```
*cdd input file : ucube.ine (6 x 4)
*cdd output file: ucube.ext
*Incidence of input (=inequalities/facets) w.r.t. output (=vertices/rays).
*row 7 is redundant;dominated by: 1 2 3 4 6
*row 6 is redundant;dominated by: 1 2
begin
  6  5  5
 -2 : 2 3
 -2 : 1 2
 -2 : 1 4
 -2 : 3 4
 -1 : 5
 2 : 4 5
 1 : 5
end
```

After "begin", there are three numbers 6 5 5. The first number 6 is a number of input (inequalities). The next number 5 is m, the number of vertices and rays in the output file. The last number 5 is always m for all *.icd files. The remaining lines can be interpreted similarly with ucube.ecd file. The input_incidence option is **not available in cdd**.

The **#incidence** option can be used when you do not wish to output the incidence file but to output only the cardinality of incidence for each output, at the end of each output line.

The incidence files (adjacency file, input_adjacency as well) can be created independently after *.ext file is created, see "postanalysis" option.

nondegenerate option

When this option is set, the program assumes that the input system is not degenerate, i.e., there is no point in the space R^d satisfying more than d inequalities of input with equality. It will run faster with this option, but of course, if this option is set for degenerate inputs, it is quite possible that the output is incorrect. The default is this option being off.

adjacency option

This option can be used when you want to output the adjacency of output. When the output is the list of vertices and rays, the program will output the adjacency list. For the example input "ucube.ine", the following extra file "ucube.ead" [2] will be created:

```
*Adjacency List of output (=vertices/rays)
*cdd input file : ucube.ine (6 x 4)
*cdd output file: ucube.ext
begin
  5
 1 3 : 2 4 5
 2 3 : 1 3 5
 3 3 : 2 4 5
 4 3 : 1 3 5
 5 4 : 1 2 3 4
end
```

The first number 5 is simply the number of outputs of cdd, the number of vertices and rays in this case. The second line 1 3 : 2 4 5 says that the first output of ucube.ext file has degree (valency) 3, and its three neighbors are 2nd, 4th and 5th output.

When the computation is to obtain the hull (inequality system), the adjacency is of course that of inequalities (i.e. facets).

The adjacency file (incidence file, input_adjacency file) can be created independently after *.ext file is created, see "postanalysis" option.

[2]In earlier versions, this was "ucube.adj"

input_adjacency option

This is for cdd+ and **not available in cdd**. This option is for outputing the adjacency of input inequalities. Here, two inequalities are defined to be *adjacent* if they are nonredundant and there is no third input inequality which is satisfied with equality at all points of the polyhedron that satisfy the two inequalities with equality. In more intuitive language, two inequalities are adjacent if each determine a facet of the polyhedron and the intersection of the two facets is not contained in any other facet.

The default file name for this output is *.iad. This file lists the redundancy information of input also. For the example "ucube.ine" above, the following "ucube.iad" will be generated:

```
*Adjacency List of input (=inequalities/facets)
*cdd input file : ucube.ine (6 x 4)
*cdd output file: ucube.ext
*row 7 is redundant;dominated by: 1 2 3 4 6
*row 6 is redundant;dominated by: 1 2
begin
  7
 1 3 : 2 4 5
 2 3 : 1 3 5
 3 3 : 2 4 5
 4 3 : 1 3 5
 5 4 : 1 2 3 4
 6 0 :
 7 0 :
end
```

Observe that the 6th inequality and the artificially added 7th inequality (infinity) are found redundant. The 7th inequality is redundant because the first four facets intersects at a single infinity point (corresponding to a unique extreme ray) and hence the polyhedron has no infinity facet, although the polyhedron is not bounded.

The input_adjacency file can be created independently after *.ext file is created, see "postanalysis" option.

While cdd+ uses both *.ine and *.ext files to compute the adjacency of input, it can be computed very efficiently by linear programming technique, using only the input data. This will be explained in the Polyhedral computation FAQ [Fuk97].

postanalysis option

It is often more desirable to compute the adjacency, input_adjacency, incidence and input_incidence relations independently from the main (and often heavy) computation of enumerating all vertices and extreme rays. The "postanalysis" option can be used together with "adjacency" and/or "incidence" options for this purpose to create *.adj and/or *.icd files from both *.ine and *.ext files. If *.ine file contains this option, cdd+ will open the corresponding *.ext file and output requested *.adj, *.iad and/or *.icd files. An error occurs when *.ext file does not exist in the current directory.

lexmin, lexmax, minindex,mincutoff, maxcutoff, mixcutoff, random options

The double description is an incremental algorithm which computes the vertices/rays of a polyhedron given by some k of original inequalities from the precomputed vertices/rays of a polyhedron given by $k-1$ inequalities. It is observed that the efficiency of the algorithm depends strongly on how one selects the ordering of inequalities, although a little can be said theoretically. These options are to select the ordering of inequalities to be added at each iteration, and it is recommended to do small experiment to select good ordering for a specific type of problems. Unfortunately, a good ordering depends on the problem and there does not seem to be THE BEST ordering for every computation. From our experiences, lexmin, lexmax, mincutoff, maxcuoff work quite well in general.

The default is lexmin ordering which simply order inequalities with respect to lexicographic ordering of rows of $(b, -A)$. The lexmax is reverse of lexmin. The mincutoff (maxcutoff) option selects an inequality which cuts off the minimum (maximum) number of vertices/rays of the $(k-1)$st polyhedron. The mixcutoff option is the mixture of mincutoff and maxcutoff which selects an inequality which cuts off the $(k-1)$st polytope as unbalanced as possible. The maxcutoff option might be efficient if the input contains many redundant inequalities (many interior points for hull computation). The minindex option selects the hyperplanes from the top of the input.

The random option selects the inequalities in a random order. This option must be followed by a random seed which is positive integer (less than 65536). For example, **random 123** specifies the random option with the random seed 123.

initbasis_at_bottom option

When this option is set, the program tries to select the initial set of rows for the double description method from the bottom of the input. This means that if the last (d+1) rows are independent, they will be chosen to initiate the algorithm.

This option is *not* default. The default follows the same ordering as the ordering of inequalities chosen. This means that if **lexmin** is the ordering of inequalities, then the initial independent rows will be chosen sequentially with lexico-min ordering. There are exceptions when this rule is not applicable, i.e. when one of mincutoff or maxcutoff options is chosen. In such cases, **lexmin** ordering will be chosen.

maximize, minimize options

When maximize option is set with an objective vector $c_0\, c_1\, c_2 \ldots c_d$, the program simply solves the linear program: $\max c_0 + c_1x_1 + c_2x_2 + \cdots + c_dx_d$ over the input polyhedron P. The grammar is simply

various comments				
H-representation				
begin				
m	$d+1$	numbertype		
b	$-A$			
end				
maximize				
c_0	c_1	c_2	$\cdots$	c_d

The minimize option works exactly same way for minimization of a linear objective function. See the sample input file “lptest.ine”. The program cdd will output both primal and dual optimal solutions if the LP is solvable. If the LP is infeasible (dual infeasible), then it will output an evidence.

For the moment, one can use either the dual simplex method (option “dual-simplex”, default) or the criss-cross method by Terlaky-Wang. The latter method can be specified by option "criss-cross" and is very sensible to the ordering of inequalities. The ordering options such as maxindex, lexmin and random will affect the behavior of this solver. Try to use a different ordering, if the computation takes too much time.

Also, in order to see the intermediate LP sign tableau one can use "show_tableau" option. Also use "manual_pivot" option to select pivots manually. Of course, these options are intended for very small problems.

The minimize and maximize options should be used only in H-representation (*.ine) files, and the output filename is "*.lps".

find_interior option

This is for cdd+ and **not available in cdd**. When this option is set, the program solves the linear program: $\max x_{d+1}$ subject to $Ax + ex_{d+1} \leq b$, where e is the column vector of all 1's. If the optimum value is zero, the polyhedron has no interior point. If the optimum value is negative then the polyhedron is empty. If the LP is dual inconsistent, then the polyhedron admits unbounded inscribed balls. To find any interior point in this last case, one must add some inequality(ies) to bound the polyhedron.

This option should be used only in H-representation (*.ine) files, and the output filename is "*.lps".

facet_listing, vertex_listing options

These are for cdd+ and **not available in cdd**. When the option "facet_listing" is set, the program checks for each i-th row of the input whether the associated inequality $A_i x \leq b_i$ determines a facet of the polyhedron. This option should be used only in H-representation (*.ine) files, and the output filename is "*.fis".

When the option "vertex_listing" is set, the program checks for each i-th row of the input whether the associated point v_i determines a vertex of the polyhedron. This option should be used only in V-representation (*.ext) files, and the output filename is "*.vis".

After *.vis or *.fis file (say test.vis) is obtained, one can get the minimal nonredundant system by using the included gawk script get_essential:

```
% get_essential <  test.vis >test\_ess.ine
```

You must have a gnu gawk command accessible at the current unix directory. One must edit the new file test_ess.ine slightly according to the instruction written in the file.

facet_listing_external, vertex_listing_external options

These options can be used to apply facet_listing and vertex_listing with a (possibly large) external file. When the option facet_listing_external is set with *.ine file, cdd+ will open an external file *.ine.external (in H-format) and verify for each inequality of the external file whether it changes the original polyhedron (represented by *.ine) if it is added. The option vertex_listing_external can be set in *.ext file and works similarly. Since cdd+ reads each line of the external file one by one, the file can be very large, say of few hundred thousands lines.

tope_listing option

This is for cdd+ and **not available in cdd**. When this option is set, the program generates all full-dimensional regions (which are sometimes called topes) of the arrangement of hyperplanes $\{h_i : i = 1, 2, \ldots, m\}$, where $h_i = \{x : A_i x \leq b_i\}$. Each tope will be represented by its location vector, i.e. a sign vector in $\{+, -\}^m$ whose i-component indicates the (positive or negative) side of the hyperplane h_i the tope is located. This procedure assumes that the input polyhedron is full-dimensional and thus the vector of all +'s determines a tope. This option should be used only in H-representation (*.ine) files, and the output filename is "*.tis".

partial_enumeration, equality, linearity, strict_inequality options

With partial_enumeration option (or equivalently equality or linearity options), one can enumerate only those vertices and rays that are lying on the set of hyperplanes associated with specified inequality numbers. If you want to compute all vertices/rays lying on hyperplanes associated with k inequalities $i_1, i_2, \ldots, i_k$ $(1 \leq i_j \leq m)$, then the option should be specified as

```
various comments
H-representation
begin
m   d+1   numbertype
 b    -A
end
partial_enumeration
k  i1  i2  ...  ik
```

The **strict_inequality** option follows the same grammar as partial_enumeration or equality. With this, cdd outputs only those vertices and rays not satisfying any of the specified inequalities with equality. See the sample files, partialtest1.ine and partialtest2.ine.

These options make no effect on LP maximization or minimization.

preprojection option

This option is for a preprocessing of orthogonal projection of the polyhedron to the subspace of R^d spanned by a subset of variables. That is, if the inequality inequality system is of two-block form $A_1 x_1 + A_2 x_2 \leq b$, and the variable indices for x_1, say $1, 4, 6, 7$, are listed in the input file as

```
H-representation
begin
m   d+1   numbertype
 b    -A
end
preprojection
4  1  4  6  7
```

Then, cdd+ will output the inequality system, $A_1 x_1 \leq b$, together with the list R of extreme rays of the homogeneous cone $\{z : z \geq 0 \text{ and } z^T A_2 = 0\}$. Consequently, the inequality system $\{ \quad r^T A_1 x_1 \leq r^T b : \quad r \in R\}$ represents the projection of the original polyhedron onto x_1-space with possible redundancy. The default file names for the inequality system output and the extremal ray output are *sub.ine and *.ext, respectively if the input file is named *.ine.

There is a supplementary C program, called domcheck, written by F. Margot, EPFL, which generates quickly a minimal (i.e. nonredundant) system from these two outputs.

This program can be obtained from the standard ftp site for cdd.

zero_tolerance option
This option is for cdd+ and **not available in cdd**. This affects only the floating-point computation with cddf+. This is to change the zero tolerance for floating point computation to a user-specified value. The default value is set in cddtype.h file as 10^{-6}. This should not be changed if you are not familiar with pitfalls of floating-point computation. A tolerance value should follow the option with blank(s) or a line break. If a tolerance is too small (e.g. 10^{-20}), or if it is too big (e.g. 0.1), the computation will most likely fail.

round_output_off, output_digits options
These options are for cdd+ and **not available in cdd**. This affects only the floating-point computation with cddf+. This is to modify the default output format of numbers. The default is to output a number with 8 digits in scientific notation, except when the nearest integer is within 10^{-6}, it outputs the integer. The first option is to cancel the latter rounding. This is recommended when zero_tolerance option above is used to change the tolerance. The second option followed by a positive integer will set the number of digits for each number to the integer.

5 How to Use

The program hardly has any user interface. Once you have compiled executable files, *cdd*, *cddf+* and *cddr+* (see Section 6 for cdd, Section 7 for cdd+), and once you create an input file, say, *test.ine*, you have basically two ways to run the program. The simplest way is just to run cdd/cdd+ with

```
% cdd test.ine
```

or

```
% cddf+ test.ine
```

or, if you want to compute with rational (exact arithmetic)

```
% cddr+ test.ine
```

Then the program will open necessary output files with default file names as shown in Table 1 and output the requested results.

If you wish to specify the output file names different from default, simply run the program by

```
% cdd   (cddf+, cddr+)
```

and input desired file names at each of file name requests. Even after you run cdd this way, one can change to the automatic mode by inputing the input file name with additional semicolon, e.g. "test.ine;".

To test cdd/cdd+, it is suggested to run cdd+ with sample input files which are stored in subdirectories **ine** and **ext**.

	Input File Format	
options	H-format (*.ine)	V-format (*.ext)
conversion	*.ext	*.ine
incidence	*.ecd	*.icd
input_incidence	*.iad	*.ead
adjacency	*.ead	*.iad
input_adjacency	*.iad	*.ead
maximize/minimize	*.lps	non applicable
facet_listing	*.fis	non applicable
vertex_listing	non applicable	*.vis
tope_listing	*.tis	non applicable
verify_input	*.solved	*.solved
preprojection	*sub.ine and *.ext	non applicable

Table 1: Default extensions for output files

6 Source Files and Compilation for cdd

(1) [Files and Compilation] The source files for distribution are

cdd.c	C main source file
cddarith.c	C sub source file
cddio.c	C sub source file
cdd.h	header file for cdd.c
cdddef.h	cdd definition file (whose two lines are to be edited by user)
dplex.c	dual simplex library
dplex.h	header file for dplex
dplexdef.h	additional header file for dplex
dplex_test.c	sample main program for dplex
setoper.c	C library for set operation
setoper.h	header file for setoper.c
cddman.tex	Latex source of this manual itself
makefile	makefile for cdd+ compilation
HISTORY.cdd	brief description of changes made at each updates
README.cdd+	cdd+ readme file
ine	A subdirectory containing sample inequality files
ext	A subdirectory containing sample points files for hull computation
COPYING	GNU GENERAL PUBLIC LICENSE

To compile the code in a standard unix environment (with GNU gcc compiler), type

```
make all
```

to obtain an optimized executable file cdd and dplex_test. If this does not work, modify the file, Makefile. The GNU compiler gcc can be replaced by cc, if aother ANSI C compiler cc is available. Since the program includes some standard ANSI library headers such as stdlib.h and time.h at compilation, the compiler must know the locations of the standard ANSI libraries. Also, the files cdd.c, cdd.h, cdddef.h, cddarith.c, cddio.c, dplex.c, dplexdef.h, dplex.h, setoper.c and setoper.h are supposed to be in the current directory.

(2) [Recompilation] The first two constants in the program dplexdef.h are to be changed by the user if necessary, and the program must be recompiled each time after any change is made. These constants are simply to specify the largest size of acceptable input data $(b, -A)$:

```
#define dp_MMAX   5002  /* USER'S CHOICE: max row size of A plus two */
#define dp_NMAX   101   /* USER'S CHOICE: max column size of A plus one */
```

If this input data has m rows and $d + 1$ columns, then in the program, dp_MMAX should be at least $m + 1$ and dp_NMAX should be at least $d + 1$. Although it has no sense to set the sizes dp_MMAX and dp_NMAX much larger than necessary, the program only creates spaces for dp_MMAX+dp_NMAX pointers and uses only necessary storage space for each input, and thus large dp_MMAX and dp_NMAX won't be too harmful.

(3) [TURBO/THINK C Users] The program cdd.c is written in ANSI C, and thus it should run on personal computers without any changes if one uses a compliler supporting ANSI standard. I have been using THINK C without any problems if ANSI library is added to the project together with cdd.c, cddarith.c, cddio.c, dplex.c and setoper.c. I could also compile it with some old version of TURBO C. Due to limited memory capacities of these compilers, it is perhaps necessary to reduce the constants MMAX and NMAX to, say 2001 and 51.

7 Source Files and Compilation for cdd+

(1) [Files and Compilation] The source files for distribution are

cdd+.readme	The readme file
cdd.C	C++ main source file
cddarith.C	C++ main arithmetic code
cddpivot.C	C++ pivot operation arithmetic code
cddio.C	C++ IO code
cddrevs.C	C++ reverse search code
cdd.h	The header file for cdd.C
cdddef.h	cdd+ definition file (whose two lines are to be edited by user)
cddtype.h	cdd+ arithmetic type definition file
cddrevs.h	The header file for cddrevs.C
setoper.C	C++ library for set operation
setoper.h	The header file for setoper.C
cddman.tex	Latex 2ϵ source file of cdd+ Reference Manual
cddman.bbl	bibliography file of cdd+ Reference Manual
cddHISTORY	brief description of changes made at each updates
ine	A subdirectory containing sample inequality input files
ext	A subdirectory containing sample point/ray input files
makefile	gcc-2.8.* makefile for cddf+ and cddr+
get_essential	gawk script for facet_listing and vertex_listing
gmp_integer.cc	Polymake's GMP wrapper in C++
gmp_rational.cc	Polymake's GMP wrapper in C++
COPYING	GNU GENERAL PUBLIC LICENSE

For compilation of cdd+, one needs either GNU's GMP (libgmp.a) or GNU's g++ Rational library (libg++.a) . It is strongly recommended to use GMP since it runs much faster.

Note that libg++ is not supported by GNU any more. Also try to use a recent GNU's C++ compiler (gcc-2.8.* or higher). Most likely you have to edit makefile according to the setup of a GNU gcc compiler, GMP or g++-library, and type

```
% make all
```

which creates two executables, cddr+ (either cddr+_gmp or cddr+_g++) and cddf+. The executable cddr+ computes with rational (exact arithmetic) and cddf+ computes with floating-point arithmetic. If you want to create only one of them, use "make cddf+" or "make cddr+". Once these executables are created one might want to remove all object files *.o by

```
% rm *.o
```

We experienced some problems with older versions of gcc. Also, be aware that gcc and g++-library that come with NEXTSTEP 3.2 have bugs in the Rational library. Please use gcc and g++lib on the newest version NEXTSTEP 3.3, or build a recent gcc and g++library on older systems.

Note that cddr+ reads Polyhedra data in integer or rational number type, while cddf+ reads data in integer, rational and real number type. When cddf+ reads integer or rational numbers, it first converts them to floating point numbers and computes with floating-point arithmetic.

(2) [Recompilation] The first two constants in the program cdddef.h are to be changed by the user if necessary, and the program must be recompiled each time after any change is made. These constants are simply to specify the largest size of acceptable input data $(b, -A)$:

```
#define MMAX    5002  /* USER'S CHOICE: max row size of A plus one */
#define NMAX    101   /* USER'S CHOICE: max column size of A plus one */
```

If this input data has m rows and $d+1$ columns, then in the program, MMAX should be at least $m+1$ and NMAX should be at least $d+1$. Although it has no sense to set the sizes MMAX and NMAX much larger than necessary, the program only creates spaces for MMAX+NMAX pointers and uses only necessary storage space for each input, and thus large MMAX and NMAX won't be too harmful.

Unlike the pascal version pdd, one can set the size MMAX as large as one wants. It is no more restricted by the SET TYPE element sizes of usual Pascal compilers.

(3) [Windows/Mac Users] In principle, cdd+ with GMP can be compiled on any platform that allows GMP and comes with modern C++ compilers with STL. Yet, I have no experiences. You might also want to try the ANSI C program cdd and cddlib instead which can be found in the same ftp site as cdd+.

8 Some Useful Tips for Usage

The computation is done by floating point arithmetic by both cdd and cddf+, and by exact rational arithmetic by cddr+. Since cddr+ runs much slower (at least for the moment), use it when you need to make sure that the output is correct.

Clearly, there is no guarantee that the programs cdd and cddf+ outputs the correct result. However they seems to work correctly for many different types of polyhedra if one carefully prepares input data files. The followings are some useful tips for input data preparation to avoid badly behaving computations with cddf+.

- In cddf+, any real value is considered as zero if its absolute value is less than 10^{-6}. Since the computation is performed with double precision arithmetic, the correctness of zero recognition depends greatly on **how accurate** the input matrix $(b - A)$ is. For example, you should never use 0.9999 for the value 1. Just use the correct value 1 as it is. Unlike many LP softwares, perturbation of data can cause some serious problems. If you want to perturb your data (e.g. right hand side) for some reason, do it with large enough constants, say of order 10^{-3}.
- If your matrix contains some irrational number, say $\sqrt{5}$, please use an approximation which is correct in at least **ten** digits, i.e. 2.236067977. See the sample input file reg600-5.ine in the ine subdirectory.
- For the same arithmetic reason, please try to scale your input matrix as even as possible by multiplying appropriate constants to some rows and columns . The program cdd does not perform any scaling before computation.

9 Bugs

- When the input system is a homogeneous system of linear inequalities (i.e. the right hand side vector b is a zero vector) and the homogeneous cone determined by the system is pointed (i.e. the origin is a vertex) , the program cdd does not output this unique vertex.

10 FTP site

An anonymous ftp site for the programs is set at:

```
ftpsite:  ftp.ifor.math.ethz.ch
directory: pub/fukuda/cdd
filenames: cdd-***.tar.gz, cdd+-***.tar.gz
```

Since the file is compressed binary file, it is necessary to use binary mode for file transfer.

11 Other Useful Codes

There are several other useful codes available for vertex enumeration and/or convex hull computation such as lrs, qhull, porta and irisa-polylib. The pointers to these codes are available at

1. lrs by D. Avis [Avi97] (C implementation of the reverse search algorithm [AF92]).
2. qhull by C.B. Barber [BDH95] (C implementation of the beneath-beyond method, see [Ede87, Mul94], which is the dual of the dd method).
3. porta by T. Christof and A. Löbel [CL97] (C implementation of the Fourier-Motzkin elimination).
4. pd by A. Marzetta [Mar97] (C implementation of the primal-dual algorithm [BFM97]).
5. Geometry Center Software List by N. Amenta [Ame].

6. Computational Geometry Pages by J. Erickson [Eri].
7. Linear Programming FAQ by R. Fourer and J. Gregory [FG97].
8. ZIB Berlin polyhedral software list:
   ftp://elib.zib-berlin.de/pub/mathprog/polyth/index.html.
9. Polyhedral Computation FAQ [Fuk97].

Acknowledgements.

I am grateful to Th. M. Liebling who provided me with an ideal opportunity to visit EPFL for the academic year 1993-1994. Without his support, the present form of this program would not have existed. There are many people who helped me to improve cdd, in particular, I am indebted to David Avis, Alain Prodon, Francois Margot, Henry Crapo, Alexander Bockmayr, David Bremner, Matthew Saltzman, Ewgenij Gawrilow and Michael Joswig.

Finally, I would like to thank both H.-J. Lüthi (ETHZ) and Th. M. Liebling (EPFL) for their continuing support for the current new development (cdd+, cddlib).

References

[ABS97] D. Avis, D. Bremner, and R. Seidel. How good are convex hull algorithms. *Computational Geometry: Theory and Applications*, 7:265–302, 1997.

[AF92] D. Avis and K. Fukuda. A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra. *Discrete Comput. Geom.*, 8:295–313, 1992.

[Ame] N. Amenta. Directory of computational geometry. http://www.geom.umn.edu/software/cglist/.

[Avi97] D. Avis. *User's Guide for lrs - Version 3.2*, 1997. available from lrs homepage ftp://mutt.cs.mcgill.ca/pub/C/lrs.html.

[BDH95] C.B. Barber, D.P. Dobkin, and H. Huhdanpaa. *qhull, Version 2.1.* The Geometry Center, Minnesota, U.S.A., 1995. program and report available from ftp://geom.umn.edu/pub/software/qhull.tar.Z.

[BFM97] D. Bremner, K. Fukuda, and A. Marzetta. Primal-dual methods for vertex and facet enumeration. In *Proc. 13th Annu. ACM Sympos. Comput. Geom.*, pages 49–56, 1997.

[CGAF94] G. Ceder, G.D. Garbulsky, D. Avis, and K. Fukuda. Ground states of a ternary fcc lattice model with nearest and next-nearest neighbor interactions. *Physical Review B*, 49(1):1–7, 1994.

[CL97] T. Christof and A. Löbel. PORTA: Polyhedron representation transformation algorithm (ver. 1.3.1), 1997. http://www.iwr.uni-heidelberg.de/iwr/comopt/soft/PORTA/readme.html.

[Ede87] H. Edelsbrunner. *Algorithms in Combinatorial Geometry.* Springer-Verlag, 1987.

[Eri] J. Erickson. Computational geometry pages, list of software libraries and codes. http://www.cs.duke.edu/ jeffe/compgeom/.

[FG97] R. Fourer and J.W. Gregory. Linear programming frequently asked questions (LP-FAQ), 1997. http://www.mcs.anl.gov/home/otc/Guide/faq/linear-programming-faq.html.

[FP96] K. Fukuda and A. Prodon. Double description method revisited. In M. Deza, R. Euler, and I. Manoussakis, editors, *Combinatorics and Computer Science*, volume 1120 of *Lecture Notes in Computer Science*, pages 91–111. Springer-Verlag, 1996. ps file available from ftp://ftp.ifor.math.ethz.ch/pub/fukuda/reports/ddrev960315.ps.gz.

[Fuk97] K. Fukuda. Polyhedral computation FAQ, 1997. both html and ps versions available from http://www.ifor.math.ethz.ch/ifor/staff/fukuda/fukuda.html.

[Gri90] V.P. Grishukhin. All facets of the cut cone for $n = 7$ are known. *Europ. J. Combin.*, 11:115–117, 1990.

[Mar97] A. Marzetta. *pd – C-implementation of the primal-dual algoirithm*, 1997. code available from http://wwwjn.inf.ethz.ch/ambros/pd.html.

[MRTT53] T.S. Motzkin, H. Raiffa, GL. Thompson, and R.M. Thrall. The double description method. In H.W. Kuhn and A.W.Tucker, editors, *Contributions to theory of games, Vol. 2*. Princeton University Press, Princeton, RI, 1953.

[Mul94] K. Mulmuley. *Computational Geometry, An Introduction Through Randamized Algorithms*. Prentice-Hall, 1994.