フリーで手に入る線形計画問題ソルバとして GLPK と cdd+を紹介する．なお，商用のものでは CPLEX が有名であり，標準的に使用されている．

1 GLPK

1.1 概要

- WEB サイト→ `http://www.gnu.org/software/glpk/glpk.html`
- GLPK (Gnu Linear Programming Kit) はモスクワのアンドリュー・マコーリンによって開発された．C で書かれたライブラリである．線形計画問題を解くための単体法と主双対内点法，および，整数計画問題を解くための分枝限定法を実装している．また，GLPSOL というスタンドアローン・ソルバも持っている．割と大きな規模の問題も解けるようである (未確認)．

1.2 簡単な使い方

スタンドアローン・ソルバ GLPSOL の UNIX 上での使い方を簡単に説明する．

1.2.1 解く問題のファイルを準備する

次のような線形計画問題を解くことを考える．

$$\begin{aligned} \text{minimize} \quad & x_1 - x_2 + x_3 \\ \text{subject to} \quad & x_1 + 2x_2 + 2x_3 \geq 6, \\ & 2x_1 + x_2 + x_3 \leq 6, \\ & x_1, x_2, x_3 \geq 0. \end{aligned}$$

この問題を記述するファイル `example1.lp` を次のように作成する．

```
\* example1.lp *\

Minimize
   value: x1 - x2 + x3

Subject To
   constraint1: x1 + 2 x2 + 2 x3 >= 6

   constraint2: 2 x1 + x2 + x3 <= 6

End
```

```
\* eof *\
```

このファイル形式は「CPLEX LP 形式」と呼ばれるもので，GLPK の標準とは異なるので，注意してもらいたい．CPLEX LP 形式において，デフォルトでは全ての変数が非負変数として扱われる．そのため，非負制約に対応する記述がファイルには存在しなくてもよい．

1.2.2 実行

次のようにコマンド入力する．

```
% glpsol --cpxlp example1.lp -o example1.out
```

そうすると次のような出力が標準出力に得られる．

```
lpx_read_cpxlp: reading problem data from ‘example1.lp’...
lpx_read_cpxlp: 2 rows, 3 columns, 6 non-zeros
lpx_read_cpxlp: 13 lines were read
lpx_simplex: original LP has 2 rows, 3 columns, 6 non-zeros
lpx_simplex: presolved LP has 2 rows, 3 columns, 6 non-zeros
lpx_adv_basis: size of triangular part = 2
      0:   objval =   0.000000000e+00   infeas =   1.000000000e+00 (0)
      1:   objval =   3.000000000e+00   infeas =   0.000000000e+00 (0)
*     1:   objval =   3.000000000e+00   infeas =   0.000000000e+00 (0)
*     3:   objval =  -6.000000000e+00   infeas =   0.000000000e+00 (0)
OPTIMAL SOLUTION FOUND
Time used:   0.0 secs
Memory used: 0.1M (83724 bytes)
lpx_print_sol: writing LP problem solution to ‘example1.out’...
```

得られた解の情報などはファイル `example1.out` **に出力した．**

```
Problem:
Rows:       2
Columns:    3
Non-zeros:  6
Status:     OPTIMAL
Objective:  value = -6 (MINimum)

   No.   Row name   St   Activity     Lower bound   Upper bound    Marginal
------ ------------ -- ------------- ------------- ------------- -------------
     1 constraint1  B             12             6
     2 constraint2  NU             6                           6            -1

   No. Column name  St   Activity     Lower bound   Upper bound    Marginal
------ ------------ -- ------------- ------------- ------------- -------------
```

```
     1 x1           NL             0             0                          3
     2 x2           B              6             0
     3 x3           NL             0             0                          2

Karush-Kuhn-Tucker optimality conditions:

KKT.PE: max.abs.err. = 0.00e+00 on row 0
        max.rel.err. = 0.00e+00 on row 0
        High quality

KKT.PB: max.abs.err. = 0.00e+00 on row 0
        max.rel.err. = 0.00e+00 on row 0
        High quality

KKT.DE: max.abs.err. = 0.00e+00 on column 0
        max.rel.err. = 0.00e+00 on column 0
        High quality

KKT.DB: max.abs.err. = 0.00e+00 on row 0
        max.rel.err. = 0.00e+00 on row 0
        High quality

End of output
```

これによって，求められた最適解が $(x_1, x_2, x_3) = (0, 6, 0)$ **であり，最適値が** -6 **であることが分かる．その他，詳しい使い方は付属のドキュメントを見てもらいたい．**

2 cdd+

2.1 概要

- WEB **サイト→** `http://www.ifor.math.ethz.ch/~fukuda/cdd_home/`
- cdd (C implementation of the Double Description Method) **はチューリッヒの福田公明によって開発された．**cdd+**は** C++**による同じソフトウェアの実装であるが，**cdd **では浮動小数点計算に基づく計算しか出来なかったのに対して，**cdd+**では有理計算に基づく計算も出来るため，例えば，有理係数を持つ線形計画問題を本当の意味で厳密に解くことが可能である．線形計画問題を解くためのアルゴリズムとしては二重記述法** (double description method) **という標準的ではないものを採用しているが，そのため，実行可能領域の端点の列挙なども可能になっている．しかし，大規模な問題を解くことはできない．**

2.2 簡単な使い方

cdd+**に付属してくるコマンドは** `cddf+`**と** `cddr+`**である．コマンド** `cddf+`**は浮動小数点計算，コマンド** `cddr+`**は有理計算に基づく計算をする．**

2.2.1 解く問題のファイルを準備する

先と同様に，次の線形計画問題を解くことを考える．

$$\begin{aligned} \text{minimize} \quad & x_1 - x_2 + x_3 \\ \text{subject to} \quad & x_1 + 2x_2 + 2x_3 \geq 6, \\ & 2x_1 + x_2 + x_3 \leq 6, \\ & x_1, x_2, x_3 \geq 0. \end{aligned}$$

この問題を記述するファイル `example1.ine` を次のように作成する．

```
* example1.ine
H-representation
begin
5 4 integer
-6  1  2  2
6  -2 -1 -1
0   1  0  0
0   0  1  0
0   0  0  1
end
minimize
0 1 -2 1
```

このファイル形式は cdd+独自のものである．

2.2.2 実行

次のようにコマンド入力する．

```
% cddf+ example1.ine
```

そうすると次のような出力が標準出力に得られる．

```
* cdd+: Double Description Method:Version 0.77(August 19, 2003)
* Copyright (C) 1999, Komei Fukuda, fukuda@ifor.math.ethz.ch
* Compiled for Floating-Point Arithmetic
------------------------------------------------
 Enumeration of all vertices and extreme rays
 of a convex polyhedron P={ x : b - A x >= 0}
 Use hull option for convex hull computation!
------------------------------------------------
input file example1.ine is open
size = 5 x 4
number type = integer
Nonhomogeneous system with  m =     5  n =     4
```

```
Open log file example1.ddl.
*Degeneracy preknowledge for computation: None (possible degeneracy)
*LP (minimization) is chosen.
*Zero tolerance = 1e-06
Open output file example1.lps.
* cdd+: Double Description Method in C++:Version 0.77(August 19, 2003)
* Copyright (C) 1999, Komei Fukuda, fukuda@ifor.math.ethz.ch
* Compiled for Floating-Point Arithmetic
*cdd LP Result
*cdd input file : example1.ine  (5 x 4)
*LP solver: Dual Simplex
*LP status: a dual pair (x, y) of optimal solutions found.
*minimization is chosen.
*Objective function is
 0 + 1 X[1] -2 X[2] + 1 X[3]
*LP status: a dual pair (x, y) of optimal solutions found.
begin
  primal_solution
  1 :  0
  2 :  6
  3 :  0
  dual_solution
  5 :  -3
  2 :  -2
  3 :  -5
  optimal_value : -12
end
*number of pivot operations = 3
*Computation starts     at Sat Jun 11 20:04:52 2005
*            terminates at Sat Jun 11 20:04:52 2005
*Total processor time = 0 seconds
*                     = 0h 0m 0s
closing the file example1.lps
closing the file example1.ddl
```

これによって，求められた最適解が $(x_1, x_2, x_3) = (0, 6, 0)$ であり，最適値が -6 であることが分かる．線形計画問題の解に関する情報がファイル `example1.lps` として，また，計算のログがファイル `example1.ddl` として自動的に作成されている．

その他，詳しい使い方は付属のドキュメントを見てもらいたい．