6回目

GLPK入門

GLPK (GNU Linear Programming Kit)

- 線形計画問題を解いてくれる
  - 決定変数の数が少なければ解析的に解ける
    - 多くなると、人の手で解くのは困難になる
- ロシアのA.O. Makhorin (mao@mai2.rcnet.ru) 氏が開発
  - ANSI C で記述
- https://www.gnu.org/software/glpk/
  - 2017.1月に glpk-4.61.tar.gz が公開されている

GLPKのインストール(Linux)

- ITC提供のワークステーションにはインストール済
- Ubuntu では、パッケージが提供されている
  - sudo apt-get –yV install glpk
  - sudo apt-get –yV install glpk-doc
  - sudo apt-get –yV install glpk-utils

  この3個で大丈夫かな
  - sudo apt-get –yV install libglpk-dev
  - sudo apt-get –yV install libglpk0
  - sudo apt-get –yV install libglpk0-dbg
    - 提供されているのは glpk-4.60以前 かもしれない

ソースファイルからのインストール

- 例えば作業用ディレクトリを /tmp とする場合
  - $ cd /tmp
  - $ zcat [glpk-4.61.tar.gzの場所] | tar xvf –
  - $ cd glpk-4.61
  - $ ./configure
  - $ make
    - エラーが無いことを確認
  - $ make check
    - エラーが無いことを確認
  - $ sudo make install
    - 多分 /usr/local/bin/glpsol と /usr/local/lib/libglpk.* がインストールされる

glpsol の起動

- glpsol (GNU Linear Programming Solver)
  - $ rehash
    - インストール後には最初におまじないが必要なことが多い
  - glpsol –v

```
[                              ]$ glpsol -v
GLPSOL: GLPK LP/MIP Solver, v4.61
Copyright (C) 2000-2017 Andrew Makhorin, Department for Applied
Informatics, Moscow Aviation Institute, Moscow, Russia. All rights
reserved. E-mail: <mao@gnu.org>.

This program has ABSOLUTELY NO WARRANTY.

This program is free software; you may re-distribute it under the terms
of the GNU General Public License version 3 or later.
[                              ]$
```

glpsol の主要オプション

- 詳細は glpsol –h で見られる
- -m filename
  - モデルファイルをfilenameから読み込む
- -d filename
  - データファイルをfilenameから読込む (モデルファイルにデータが記述されていたら無視される)
- -y filename
  - 画面出力を filename に書き出す
- -o filename
  - 実行結果を filename に書き出す

モデルファイルの書き方(1)

- 目的関数　$\max \quad x_1 + x_2$
- 制約条件　$5x_1 + 3x_2 \le 15$

$x_1 - x_2 \le 2$

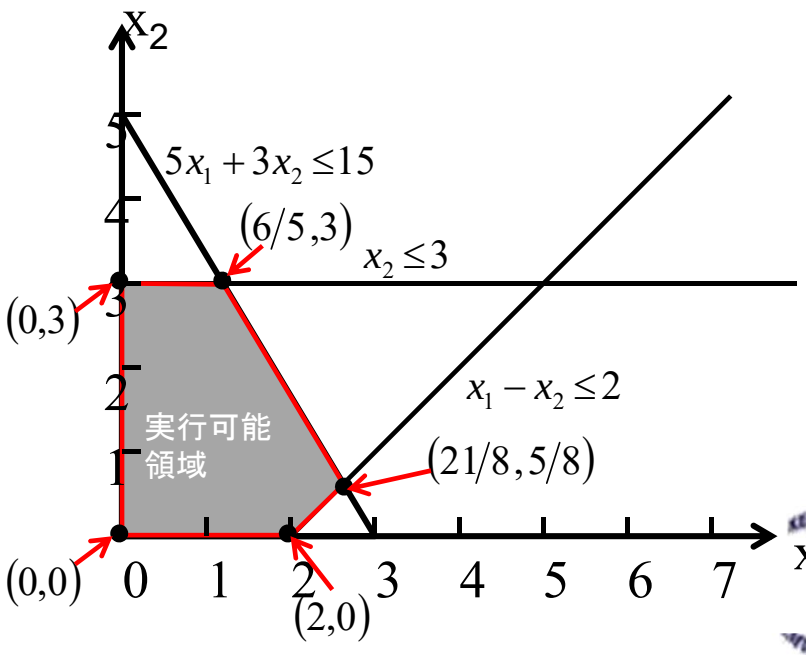
$x_2 \le 3$

$x_1 \ge 0$

$x_2 \ge 0$

$$z = x_1 + x_2$$

$$x_2 = -x_1 + z$$



モデルファイルの書き方(2)

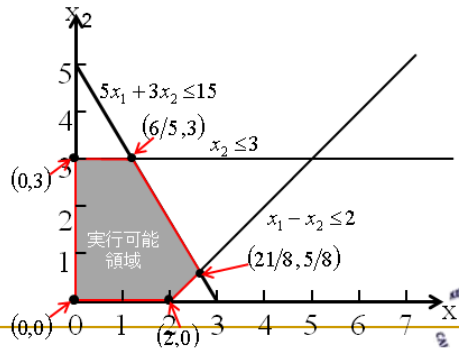
- 変数の宣言
  - var 変数名 型指定 ;
    - var x1 >= 0 ;
- 目的関数の定義
  - maximize z1: x1 + 3.5e4*x2 ;
  - minimize z2: x1 – 4.2*x2 ;
- 制約条件の定義 (subject to)
  - s.t. st1: 5*x1 + 3*x2 <= 15 ;
- モデルファイルの終了宣言
  - end ;

モデルファイルの例

- 目的関数 $\max \quad x_1 + x_2$
- 制約条件 $5x_1 + 3x_2 \le 15$
  $x_1 - x_2 \le 2$
  $x_2 \le 3$
  $x_1 \ge 0$
  $x_2 \ge 0$

$$z = x_1 + x_2$$

$$x_2 = -x_1 + z$$




```
/* lp-ex1.mod */

var x1 >= 0;
var x2 >= 0;

maximize z: x1 + x2 ;

s.t. st1: 5*x1 + 3*x2 <= 15 ;
s.t. st2: x1 - x2 <= 2;
s.t. st3: x2 <= 3 ;

end ;
```

モデルファイルを解いてみる

- $ glpsol –m lp-ex1.mod –o lp-ex1.out

```
$ glpsol -m lp-ex1.mod -o lp-ex1.out
GLPSOL: GLPK LP/MIP Solver, v4.61
Parameter(s) specified in the command line:
 -m lp-ex1.mod -o lp-ex1.out
Reading model section from lp-ex1.mod...
13 lines were read
Generating z...
Generating st1...
Generating st2...
Generating st3...
Model has been successfully generated
GLPK Simplex Optimizer, v4.61
4 rows, 2 columns, 7 non-zeros
Preprocessing...
2 rows, 2 columns, 4 non-zeros
Scaling...
 A: min|aij| =  1.000e+00  max|aij| =  5.000e+00  ratio =  5.000e+00
Problem data seem to be well scaled
Constructing initial basis...
Size of triangular part is 2
*     0: obj =  -0.000000000e+00 inf =   0.000e+00 (2)
*     3: obj =   4.200000000e+00 inf =   0.000e+00 (0)
OPTIMAL LP SOLUTION FOUND
Time used:   0.0 secs
Memory used: 0.1 Mb (102259 bytes)
Writing basic solution to 'lp-ex1.out'...
$
```

問題として正しく理解できた

最適解が見つかった

実行結果を見て見る(1)

```
$ cat lp-ex1.out
Problem:    lp
Rows:       4
Columns:    2
Non-zeros:  7
Status:     OPTIMAL
Objective:  z = 4.2 (MAXimum)

  No.   Row name   St   Activity     Lower bound   Upper bound    Marginal
------ ------------ -- ------------ ------------- ------------- -------------
     1 z            B           4.2
     2 st1          NU           15                         15           0.2
     3 st2          B          -1.8                         2
     4 st3          NU            3                         3           0.4

  No. Column name  St   Activity     Lower bound   Upper bound    Marginal
------ ------------ -- ------------ ------------- ------------- -------------
     1 x1           B           1.2             0
     2 x2           B             3             0

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
```

重要

何に使うのか
よくわからない

実行結果を見てみる(2)

```
$ cat lp-ex1.out
Problem:    lp
Rows:       4
Columns:    2
Non-zeros:  7
Status:     OPTIMAL
Objective:  z = 4.2 (MAXimum)

   No.   Row name   St   Activity     Lower bound   Upper bound    Marginal
------ ------------ -- ------------ ------------- ------------- -------------
     1 z            B           4.2
     2 st1          NU           15                         15           0.2
     3 st2          B          -1.8                         2
     4 st3          NU           3                          3           0.4

   No. Column name  St   Activity     Lower bound   Upper bound    Marginal
------ ------------ -- ------------ ------------- ------------- -------------
     1 x1           B           1.2             0
     2 x2           B             3             0
```

結果の最大値

目的関数
制約条件

最適なx1とx2の値

その他の記述の仕方 (1)

- 変数の範囲を示す
  - var x1 <=2, >=0 ;
  - var x2 >= -3, <=5 ;

$$0 \le x_1 \le 2$$

$$-3 \le x_2 \le 5$$

- 変数を整数に限定したい
  - param N interger, >0 ;　← 型指定
  - set V := 1..N ;　← Vは1～Nまでの整数
  - set E within {V,V} ;　← Eは[1～N][1～N]の二次元ベクトルの要素
  - set W := 1..N ;
  - set W1 := 1..N-1 ;　← W1は1～N-1までの整数
  - param AM{E} ;　← AMは二次元配列 AM[x, y]

その他の記述の仕方(2)

- 0, 1 の二値を取る変数
  - var y0 binary ; ← 型指定 ← y0は0 又は 1 の値を取る
  - var y{W} binary ; ← 配列y[W]は0 又は 1 の値を取る
  - var x{V, W} binary ; ← 配列x[V, W]は0 又は 1 の値を取る
- Σ
  - sum{i in W} (y[i]) ; ← $\sum_{i=1}^{N} y[i]$
- 配列の初期化 (制約条件で実行)
  - s.t. XX{v1 in V, v2 in V: AM[v1,v2]=1} : 式

文字式を利用した定式化

- モデルファイルとデータファイルを分離
  - モデルファイルは、パラメータで一般化
  - データファイルでパラメータに値を付与
    - glpsol –m filename.mod –d filename.dat –o filename.out

目的関数　$\min \sum_{(i,j)\in E} cost_{ij} x_{ij}$

※p != q が仮定されている

制約条件　$\sum_{j\in V} x_{ij} - \sum_{j\in V} x_{ji} = 1 \quad (i = p \text{のとき})$

$\sum_{j\in V} x_{ij} - \sum_{j\in V} x_{ji} = 0 \quad (\forall i \neq p,q \in V)$

$0 \le x_{ij} \le 1 \quad (\forall (i,j) \in E)$

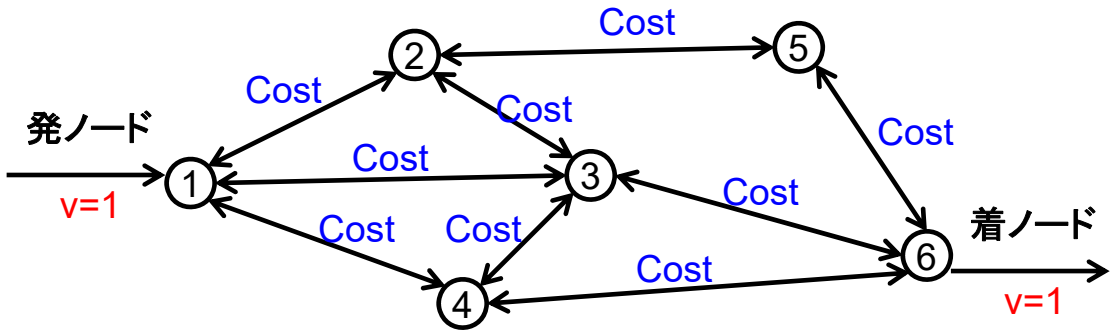
最短経路問題

- ノード p から q までの最短経路を求める
  - リンクに距離（ $Cost_{ij}$ ）を与える
- ノード p に流量 1 のトラヒックを流す
  - x_{ij} は、リンクijの流量 ⇒ 最短経路以外は 0 になる

$$\min \sum_{(i,j)\in E} cost_{ij} x_{ij}$$

$$\sum_{j\in V} x_{ij} - \sum_{j\in V} x_{ji} = 1 \quad (i = p \text{のとき})$$

$$\sum_{j\in V} x_{ij} - \sum_{j\in V} x_{ji} = 0 \quad (\forall i \neq p, q \in V)$$



Science and Technology

モデルファイル

```
/* sp-gen.mod */

/* Given parameters */
param N integer, >0;
param p integer, >0;
param q integer, >0;

set V := 1..N;
set E within {V,V};

param cost{E};

/* Decision variables */
var x{E} <=1, >=0;

/* Objective function */
minimize PATH_COST: sum{i in V} (sum{j in V} (cost[i,j]*x[i,j]) );

/* Constraints */
s.t. SOURCE{i in V: i = p && p != q}:
     sum{j in V} (x[i,j]) - sum{j in V} (x[j,i]) = 1;
s.t. INTERNAL{i in V: i != p && i != q && p != q}:
     sum{j in V} (x[i,j]) - sum{j in V} (x[j,i]) = 0;

end;
```

目的関数　$\min \sum_{(i,j)\in E} cost_{ij} x_{ij}$

制約条件　$\sum_{j\in V} x_{ij} - \sum_{j\in V} x_{ji} = 1 \quad (i = p \text{のとき})$

$\sum_{j\in V} x_{ij} - \sum_{j\in V} x_{ji} = 0 \quad (\forall i \neq p, q \in V)$

$0 \le x_{ij} \le 1 \quad (\forall (i,j) \in E)$

データファイル

```
/* sp-gen1.dat */

param p := 1;
param q := 4;
param N := 4;

param : E : cost :=
1 1 100000
1 2 5
1 3 8
1 4 100000
2 1 100000
2 2 100000
2 3 2
2 4 7
3 1 100000
3 2 100000
3 3 100000
3 4 4
4 1 100000
4 2 100000
4 3 100000
4 4 100000
;

end;
```

発ノード

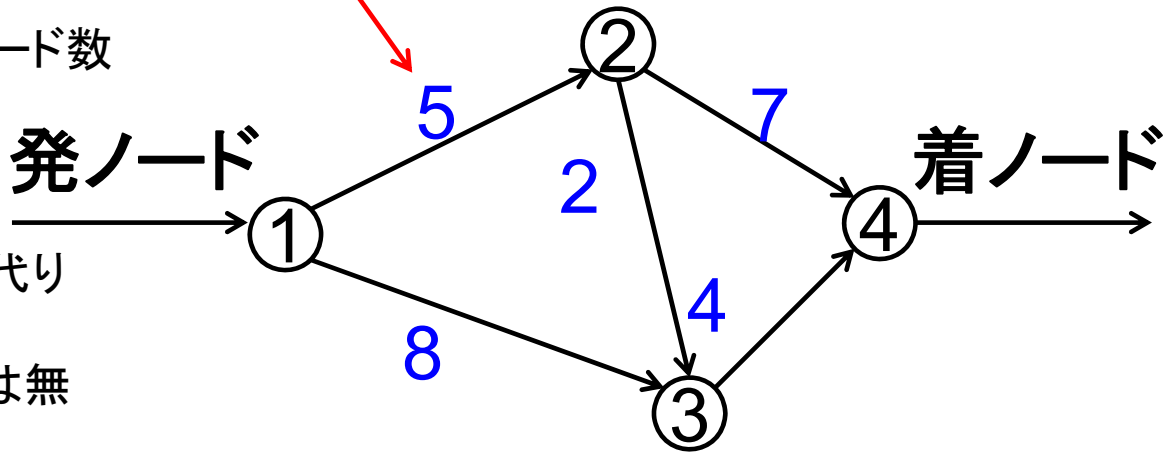
着ノード

ノード数

無限大の代り

逆向きは無

距離



最短経路は、1→2→3→4 で 11

実行結果

```
Problem:    sp
Rows:       4
Columns:    16
Non-zeros:  34
Status:     OPTIMAL
Objective:  PATH_COST = 11 (MINimum)
```

最小距離11

No.	Row name	St	Activity	Lower bound	Upper bound	Marginal
1	PATH_COST	B	11			
2	SOURCE[1]	NS	1	1	=	11
3	INTERNAL[2]	NS	0	-0	=	6
4	INTERNAL[3]	NS	0	-0	=	4

No.	Column name	St	Activity	Lower bound	Upper bound	Marginal
1	x[1,1]	NL	0	0	1	100000
2	x[1,2]	B	1	0	1	
3	x[1,3]	NL	0	0	1	1
4	x[1,4]	NL	0	0	1	99989
5	x[2,1]	NL	0	0	1	100005
6	x[2,2]	NL	0	0	1	100000
7	x[2,3]	B	1	0	1	
8	x[2,4]	NL	0	0	1	1
9	x[3,1]	NL	0	0	1	100007
10	x[3,2]	NL	0	0	1	100002
11	x[3,3]	NL	0	0	1	100000
12	x[3,4]	B	1	0	1	
13	x[4,1]	NL	0	0	1	100011
14	x[4,2]	NL	0	0	1	100006
15	x[4,3]	NL	0	0	1	100004
16	x[4,4]	NL	0	0	1	100000

演習7

整数線形計画問題

目的関数

制約条件

- 次回から二回は12-107 ワークステーション室
  - 使ってみよう GLPK