最適化・シミュレーション演習 補足 切除平面法

1. Knapsack Problem

次のナップサック問題（KP）を考える。品物 1,2,3 の値段はそれぞれ 7,8,3(万円)、重さはそれぞれ 3,4,2(kg)である。重量制限 6 kgのナップサックに入る品物のうち、最も値段の和が大きくなるような組合せを求めよ。

決定変数は以下のように定義される。

x_j =0, 品物 j をナップサックに入れる

x_j =1, 品物 j をナップサックに入れない

(KP) 最大化 $z = 7x_1 + 8x_2 + 3x_3$

制約 $3x_1 + 4x_2 + 2x_3 \leqq 6, \quad x_j \in \{0,1\}, \quad \forall\ j$

最適解は $x_1 = 0, \quad x_2 = 1, \quad x_3 = 1$ 最適目的関数値 $z = 11$

以下のモデルファイル knapsack-cut.mod では 2 つの問題#Master Problem と#Cut Generation が記述されていることに注意せよ。

```
#knapsack-cut.mod
#Master Problem
param Nitem;
param Ncut;   #妥当不等式の貸す
param weight{i in 1..Nitem};
param value{i in 1..Nitem};
param capacity;
param acut {i in 1..Ncut, j in 1..Nitem}; #妥当不等式係数
param bcut {i in 1..Ncut};                #妥当不等式定数

var x{i in 1..Nitem} binary;
maximize Profit: sum{i in 1..Nitem}value[i]*x[i];
subject to Capacity_Constraint:  sum{i in 1..Nitem}weight[i]*x[i] <= capacity;
subject to Valid {i in 1..Ncut}: sum{j in 1..Nitem} acut[i,j]*x[j] <=bcut[i];

#Cut Generation
param xx{i in 1..Nitem};
var y{i in 1..Nitem} binary;

maximize Violation: sum{i in 1..Nitem}(xx[i]-1)*y[i];
subject to Cover: sum{i in 1..Nitem}weight[i]*y[i] >=capacity+0.001;
```

データは以下の通りである。

```
param Nitem:=3;
param:value weight:=
1 7 3
2 8 4
3 3 2;
param capacity:=6;
```

2. 妥当不等式と分離問題

```
model knapsack-cut.mod ;
data knapsack.dat ;
option display_round 6;
option solver cplexamp;

problem knapsack_master: x, Profit, Capacity_Constraint, Upper_Lower_Bound, Valid;
   option relax_integrality 1; #Master 問題は緩和 LP を解く
problem cut_generaton: y, Violation, Cover;
   option relax_integrality 0; #妥当不等式生成問題は、厳密解を求める。

let Ncut:=0;
repeat
{
   solve knapsack_master;
   expand > knapsack.sol;
   display Profit > knapsack.sol;
   display x > knapsack.sol;
   for{i in 1..Nitem}
   {
        let xx[i]:=x[i];  #緩和 Master 問題の最適解を妥当不等式生成問題の係数とする。
   }
   solve cut_generaton;   #妥当不等式生成問題を解く。
   expand > knapsack.sol;
   display Violation > knapsack.sol;
   display y > knapsack.sol;

   if (Violation > -1) then  #切除平面が生成されるならば
   {
       let Ncut:=Ncut+1;     #妥当不等式の本数を増加。
       for{j in 1..Nitem}    #妥当不等式の係数を定める。
       {
          let  acut[Ncut, j]:=y[j];
       }
          let bcut[Ncut]:=sum{j in 1..Nitem}y[j]-1;
   }
   else
   {
        break;  #切除平面が生成されなければ、repeat 反復より脱出。
   }
}

  option knapsack_master.relax_integrality 0;  #Master 問題の厳密解を求める。
  solve knapsack_master;
```

3. 実行結果

```
C:¥Document¥MyDocument¥最適化シミュレーション演習 2014¥補足 切除平面¥KP-cut>ampl
knapsack-cut.run
AMPL Version 20080423 (x64_win64)
CPLEX 12.4.0.0: optimal solution; objective 13   #Master 緩和問題 (1 回目)
1 dual simplex iterations (0 in phase I)
CPLEX 12.4.0.0: optimal integer solution; objective -0.25  #妥当不等式生成問題 (1 回目)
0 MIP simplex iterations
0 branch-and-bound nodes
CPLEX 12.4.0.0: optimal solution; objective 11  #Master 緩和問題 (2 回目)
1 dual simplex iterations (0 in phase I)
Solution determined by presolve;                #妥当不等式生成問題 (2 回目)
objective Violation = -1.                       シャープカット生成できなかった。
CPLEX 12.4.0.0: optimal integer solution; objective 11 #Master 問題 (厳密解)
0 MIP simplex iterations
0 branch-and-bound nodes
```

以下 sol ファイルの内容を示す。

```
#Master 緩和問題 (1 回目)
maximize Profit:
        7*x[1] + 8*x[2] + 3*x[3];

subject to Capacity_Constraint:
        3*x[1] + 4*x[2] + 2*x[3] <= 6;

subject to Upper_Lower_Bound[1]:
        0 <= x[1] <= 1;

subject to Upper_Lower_Bound[2]:
        0 <= x[2] <= 1;

subject to Upper_Lower_Bound[3]:
        0 <= x[3] <= 1;

Profit = 13.000000

x [*] :=
1  1.000000
2  0.750000
3  0.000000
;
#妥当不等式生成問題 (1 回目)
maximize Violation:
        -0.25*y[2] - y[3];

subject to Cover:
        3*y[1] + 4*y[2] + 2*y[3] >= 6.001;
```

```
Violation = -0.250000

y [*] :=
1  1.000000
2  1.000000
3  0.000000
;
#Master 緩和問題 (2 回目)
maximize Profit:
        7*x[1] + 8*x[2] + 3*x[3];

subject to Capacity_Constraint:
        3*x[1] + 4*x[2] + 2*x[3] <= 6;

subject to Upper_Lower_Bound[1]:
        0 <= x[1] <= 1;

subject to Upper_Lower_Bound[2]:
        0 <= x[2] <= 1;

subject to Upper_Lower_Bound[3]:
        0 <= x[3] <= 1;

subject to Valid[1]:
        x[1] + x[2] <= 1;

Profit = 11.000000

x [*] :=
1  0.000000
2  1.000000
3  1.000000
;
#妥当不等式生成問題 (2 回目)
maximize Violation:
        -y[1];

subject to Cover:
        3*y[1] + 4*y[2] + 2*y[3] >= 6.001;

Violation = -1.000000  #カット生成できなかった。

y [*] :=
1  1.000000
2  1.000000
3  0.000000
;
```