

最適化・シミュレーション演習 第2回 数理計画問題の定式化

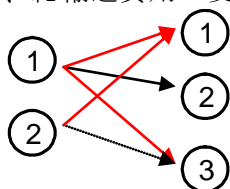
1. 輸送問題 (イントロダクション)

工場2か所から倉庫3か所への輸送問題を考える。工場 i から倉庫 j への単位輸送量あたりの輸送コスト c_{ij} ($i=1, \dots, 2, j=1, \dots, 3$), 工場 i からの供給量 S_i ($i=1, \dots, 2$), 倉庫 j の需要量 D_j ($j=1, \dots, 3$)は表のように与えられている。最小費用となる各工場から各倉庫への輸送量を求める。

(1) ハウザッカー法により実行可能解を求め、表に輸送量を記入し、総輸送費用を求めよ。

倉庫 工場	倉庫 1	倉庫 2	倉庫 3	供給量
工場 1	10 6	15 5	10 4	35
工場 2	15 8	0 9	0 5	15
需要量	25	15	10	

(2) 輸送量 0 の辺で最も輸送コストが安い辺を 1 つ選ぶと、輸送量が正の辺と閉路 (ループ) をなす。この閉路を以下のグラフに記入せよ。またこのとき選んだ輸送量が 0 の辺に、輸送量を θ (≥ 0) 追加する場合、総輸送費用の変動量を θ の関数として表せ。



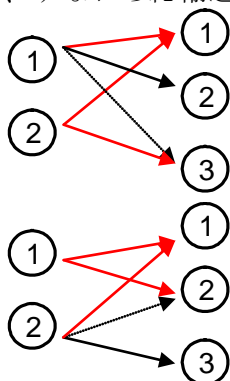
$$\begin{aligned}
 & \text{総輸送費用の変動量(辺 23 を選択)} \\
 & = (\theta \text{ 追加前の総輸送費用} - \theta \text{ 追加後の総輸送費用}) \\
 & = \underline{(c_{23} - c_{13} + c_{11} - c_{21}) \theta = (5 - 4 + 6 - 8) \times \theta = -\theta}
 \end{aligned}$$

(3) 問題(2)のループの各辺における輸送量が非負であるという条件を用いて、最も輸送費用が少なくなる θ の値を求めよ。このように求められた θ を用いて変更された輸送量を以下の表に記入し、更新された総輸送費用を求めよ。

θ の値 工場	倉庫 1	倉庫 2	倉庫 3	供給量
10	20 6	15 5	0 4	35
総輸送費用	5 8	0 9	10 5	15
285	需要量	25	15	10

(4) 問題 1(3)で更新され

た解において、新たに輸送量が 0 となる辺を選ぶと問題 1(3)と同様に閉路が得られる。この閉路を以下の図に示せ。また、どの辺を選んで輸送量を θ (≥ 0) 追加させても、これ以上総輸送費用は改善できないこと、すなわち総輸送費用の変動量が正であることを示せ。



$$\begin{aligned}
 & \text{総輸送費用の変動量(辺 13 を選択)} \\
 & = (\theta \text{ 追加前の総輸送費用} - \theta \text{ 追加後の総輸送費用}) \\
 & = \underline{(c_{13} - c_{23} + c_{21} - c_{11}) \theta = (4 - 5 + 8 - 6) \times \theta = \theta} \\
 & \text{総輸送費用の変動量(辺 22 を選択)} \\
 & = (\theta \text{ 追加前の総輸送費用} - \theta \text{ 追加後の総輸送費用}) \\
 & = \underline{(c_{22} - c_{12} + c_{11} - c_{21}) \theta = (9 - 5 + 6 - 8) \times \theta = 2\theta}
 \end{aligned}$$

また、輸送問題を次のように線形計画問題として定式化する。

$$\text{目的関数 } \min z = 6x_{11} + 5x_{12} + 4x_{13} + 8x_{21} + 9x_{22} + 5x_{23}$$

$$\text{制約条件(1-1)} \quad x_{11} + x_{12} + x_{13} = 35$$

$$\text{制約条件(1-2)} \quad x_{21} + x_{22} + x_{23} = 15$$

$$\text{制約条件(2-1)} \quad x_{11} + x_{21} = 25$$

$$\text{制約条件(2-2)} \quad x_{12} + x_{22} = 15$$

$$\text{制約条件(2-3)} \quad x_{13} + x_{23} = 10$$

$$\text{非負条件 } x_{ij} \geq 0, i=1,2, j=1,\dots,3$$

この問題を基底形式に変形するため、目的関数から x_{11} , x_{12} , x_{13} , x_{21} を消去し、制約条件(1-1)を取り除く。また、制約(2-1)から制約(1-2)を引く。この操作により次の基底形式の線形計画問題が得られる。

$$\begin{aligned} \text{目的関数 } \min z &= 6(10 + x_{22} + x_{23}) + 5(15 - x_{22}) + 4(10 - x_{23}) + 8(15 - x_{22} - x_{23}) + 9x_{22} + 5x_{23} \\ &= 2x_{22} - x_{23} + 295 \end{aligned}$$

$$\text{制約条件(1-2)} \quad x_{21} + x_{22} + x_{23} = 15$$

$$\text{制約条件(2-1)} \quad x_{11} - x_{22} - x_{23} = 10$$

$$\text{制約条件(2-2)} \quad x_{12} + x_{22} = 15$$

$$\text{制約条件(2-3)} \quad x_{13} + x_{23} = 10$$

$$\text{非負条件 } x_{ij} \geq 0, i=1,2, j=1,\dots,3$$

この問題を単体法により解け。

基底変数	値	z	x_{11}	x_{12}	x_{13}	x_{21}	x_{22}	x_{23}	θ
z	295	1	0	0	0	0	-2	1	
x_{21}	15	0	0	0	0	1	1	1	15
x_{11}	10	0	1	0	0	0	-1	-1	
x_{12}	15	0	0	1	0	0	1	0	
x_{13}	10	0	0	0	1	0	0	1	10
基底変数	値	z	x_{11}	x_{12}	x_{13}	x_{21}	x_{22}	x_{23}	θ
z									

最適目的関数値 $z =$ _____.

最適解 $x_{11} =$ _____, $x_{12} =$ _____, $x_{13} =$ _____, $x_{21} =$ _____, $x_{22} =$ _____, $x_{23} =$ _____.

#houthakker.run

```
param NPLANT;      #工場数
param NDEMAND;     #倉庫数
param c {i in 1..NPLANT, j in 1..NDEMAND}>=0; #輸送コスト
param x {i in 1..NPLANT, j in 1..NDEMAND}>=0; #輸送量
param S {i in 1..NPLANT}>=0; #供給量
param D {j in 1..NDEMAND}>=0; #需要量
param newc {k in 1..NPLANT*NDEMAND}>=0; #安い順に並べた費用
param n_pl {k in 1..NPLANT*NDEMAND};      #並べ替えた費用(k 番目)に対応する工場
param n_dem {k in 1..NPLANT*NDEMAND};     #並べ替えた費用(k 番目)に対応する倉庫
param tot_num; #並べ替えた費用の順番
param tmp; #変数内容入れ替えのためのパラメタ
data tp.dat; #データファイルの読み込み
let tot_num:=1;
for{i in 1..NPLANT}  #c[i,j] を new_c[tot_num]に入れる、tot_num=1,..., NPLANT*NDEMAND
{
    for{j in 1..NDEMAND}
    {
        let newc[tot_num]:=c[i,j];
        let n_pl[tot_num]:=i;
        let n_dem[tot_num]:=j;
        let tot_num:=tot_num+1;
        if(j = NDEMAND) then #c[1,1],c[1,2],c[1,3]まで new_c[1]から new_c[3]に入れ、次は c[2,1]
        {
            break;
        }
    }
}
for{i in 0..NPLANT*NDEMAND-2} # new_c[tot_num] を昇順に並べ替え
{
    for{j in 1..NPLANT*NDEMAND-1-i}
    {
        if(newc[j]>newc[j+1])then
        {
            let tmp:=newc[j]; let newc[j]:=newc[j+1]; let newc[j+1]:=tmp;
            let tmp:=n_pl[j]; let n_pl[j]:=n_pl[j+1]; let n_pl[j+1]:=tmp;
            let tmp:=n_dem[j]; let n_dem[j]:=n_dem[j+1]; let n_dem[j+1]:=tmp;
        }
    }
}
let tot_num:=1;
repeat while((sum{i in 1..NPLANT}S[i]>0) and (sum{j in 1..NDEMAND}D[j]>0)
and (tot_num <=NPLANT*NDEMAND)) #供給または需要の未達成量がある限り
{
    if(min(S[n_pl[tot_num]], D[n_dem[tot_num]]) >0)then 供給または需要の未達成量の小さい値
    {#min(a,b)は a と b の小さい方を与える
        let (空白 輸送量 x の更新 );
        let (空白 供給未達成量 S の更新 );
        let (空白 需要未達成量 D の更新 );
    }
    let tot_num:= tot_num+1;
}
```

```
display x;  
exit;
```

2. 輸送問題の AMPL による定式化：モデルファイルとデータファイルの分離

次のような 3 つのファイルを作業フォルダ AMPLWORK (例えば C:\ampleml\amplwork) に作成する。

tp.mod ファイル：輸送問題を記述したファイル

tp.dat ファイル：輸送問題のデータを記述したファイル

tp.run ファイル：AMPL のコマンドスクリプトを記述したファイル

以下に内容を記す。ただし「#」以降はコメントである。

ファイル tp.mod の内容

```
param NPLANT;      #工場数定義  
param NDEMAND;    #倉庫数定義  
param c {i in 1..NPLANT, j in 1..NDEMAND} >= 0; #輸送費用  
param S {i in 1..NPLANT} >= 0;                #供給量  
param D {j in 1..NDEMAND} >= 0;                #需要量  
var x {i in 1..NPLANT, j in 1..NDEMAND} >= 0; #変数：輸送量  
minimize totalcost: sum {i in 1..NPLANT, j in 1..NDEMAND} c[i,j]*x[i,j]; #目的関数：輸送費用  
subject to supply {i in 1..NPLANT}: sum {j in 1..NDEMAND} x[i,j] = S[i]; #供給量制約  
subject to demand {j in 1..NDEMAND}: sum {i in 1..NPLANT} x[i,j] = D[j]; #需要制約
```

ファイル tp.dat の内容

```
param NPLANT := 2;      #工場数設定  
param NDEMAND := 3;     #倉庫数設定  
param c :=              #添字を有するパラメタの記述は自由度が高い。テキストを参照。  
[1,*] 1 6 2 5 3 4  
[2,*] 1 8 2 9 3 5;  
param S := 1 35 2 15;  
param D := 1 25 2 15 3 10;
```

ファイル tp.run の内容

```
model tp.mod;  
data tp.dat;  
option solver cplexamp; #ソルバに CPLEX を指定、これがないと MINOS が動く  
option display_round 6; #小数点以下 6 桁表示  
solve; #問題の求解  
expand > tp.sol; #ファイルに定式化を展開して表示  
display totalcost > tp.sol;  
display supply.dual > tp.sol; #供給制約の双対変数  
display demand.dual > tp.sol; #需要制約の双対変数  
display x > tp.sol;  
display _ampl_time > tp.sol;  
display _total_solve_time > tp.sol;  
exit;
```

3. 施設配置問題の AMPL による定式化

関東地方を中心に営業を行ってきた輸入品販売業のある会社は、関西地方に活動を拡大するため、京阪神地方に倉庫の賃借を行う計画を立てている。賃借の候補となる倉庫は m カ所にあつて、第 i 地点の倉庫 ($i=1, \dots, m$) の月間処理能力は a_i (トン/月) で、その経費 (賃借料や維持費など毎月の固定費用) は d_i (千円/月) である。また、関西一円に広がる消費地 j ($j=1, \dots, n$) での輸入品の需要量 b_j (トン/月) と、倉庫 i から需要地 j へのトン当たり輸送費 c_{ij} (千円) が与えられている。すべての需要を満たし、毎月の総費用 (倉庫経費 + 輸送費) を最小にする倉庫配置と輸送計画を求めたい。この問題を数理計画問題として定式化せよ。

$$\begin{aligned} \min & \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} + \sum_{i=1}^m d_i y_i \\ \text{subject to} & \sum_{i=1}^m x_{ij} \geq b_j, j = 1, \dots, n \\ & \sum_{j=1}^n x_{ij} \leq a_i y_i, i = 1, \dots, m \\ & x_{ij} \geq 0, i = 1, \dots, m, j = 1, \dots, n \\ & y_i \in \{0, 1\}, i = 1, \dots, m \end{aligned}$$

```
#fl.mod;
param NPLANT ;
param NDEMAND ;
var y {i in 1..NPLANT, j in 1..NDEMAND} >=0;
var x {i in 1..NPLANT} binary;
param varcost {i in 1..NPLANT, j in 1..NDEMAND} >=0;
param fixcost {i in 1..NPLANT} >=0;
param caplimit {i in 1..NPLANT} >=0;
param demand {j in 1..NDEMAND} >=0;
minimize cost:
sum {i in 1..NPLANT} fixcost[i]*x[i]
+sum {i in 1..NPLANT, j in 1..NDEMAND}
varcost[i, j]*y[i, j];
subject to capacity {i in 1..NPLANT}:
    sum {j in 1..NDEMAND} y[i, j] <= caplimit[i]*x[i];
subject to demconst {j in 1..NDEMAND}:
    sum {i in 1..NPLANT} y[i, j] = demand[j];
```

```
#fl.dat
param NPLANT := 2;
param NDEMAND :=3;
param varcost :=
[1,*] 1 72 2 9 3 34
[2,*] 1 85 2 78 3 79 ;
param fixcost :=
1 1500 2 1400 ;
param caplimit :=
1 96 2 74 ;
param demand :=
1 10 2 28 3 21 ;
```

```
#fl.run;
model fl.mod ;
data fl.dat ;
option solver cplexamp;
option display_round 6;
solve;
display cost > fl.sol;
display sum {i in 1..NPLANT} fixcost[i]*x[i] > fl.sol;
display sum {i in 1..NPLANT, j in 1..NDEMAND} varcost[i, j]*y[i, j] >
fl.sol;
display x > fl.sol;
display y > fl.sol;
quit;
```

4. 集合被覆問題の応用

あるコンサートをアリーナで開催することになった。そのコンサートは熱狂的なファンが殺到することで知られており、アリーナのコンサートも事故防止のために厳重な警備が必要と考えられる。アリーナでは、どこに警備員を配置したらよいかを検討することにした。会場担当者は、会場全体を小さなブロック（たとえば、ブロック 1,2,..., m ）に分け、仮に警備員をある候補地点（たとえば、候補地点 1,2,..., n ）に配置したときに警備員がどのブロックを同時にみることができるかを調べた。（例： $m=5, n=5$ の場合、地点 1 よりブロック 1,2 を監視可能）この情報をもとに、会場全体を最小人数の警備員で警備するための配置とそのときの警備要員の必要人数を求めたい。

候補地点	1	2	3	4	5
ブロック1	○		○		
ブロック2	○	○		○	○
ブロック3		○			○
ブロック4			○	○	
ブロック5					○

問題データ： 配置候補地点 $(1,2,...,n)$ 、警備すべきブロック $(1,2,...,m)$ ；ブロック i が候補地点 j から監視可能なら $a_{ij}=1$ （さもなければ 0）とする制約係数行列 $A=[a_{ij}]$

変数（=列）の定義： x_j =警備員を地点 j に配置するとき 1、さもなければ 0

目的関数：（警備員を配置する地点数最小）

制約条件：（各ブロックは監視可能）

```
#sc.mod
param NBLOCK ;
param NPOINT ;
param a {i in 1..NBLOCK, j in 1..NPOINT} binary;
var x {j in 1..NPOINT} binary;
minimize totalcost: sum {j in 1..NPOINT} x[j];
subject to watch {i in 1..NBLOCK}:
sum {j in 1..NPOINT} a[i, j]*x[j] >= 1;
```

```
#sc.dat
param NPOINT := 5;
param NBLOCK := 5;
param a :=
1 1 1 1 2 0 1 3 1 1 4 0 1 5 0
2 1 1 2 2 1 2 3 0 2 4 1 2 5 1
3 1 0 3 2 1 3 3 0 3 4 0 3 5 1
4 1 0 4 2 0 4 3 1 4 4 1 4 5 0
5 1 0 5 2 0 5 3 0 5 4 0 5 5 1;
```

```
#sc.run
model sc.mod ;
data sc.dat ;
option solver cplexamp;
option display_round 6;
solve;
expand > sc.sol;
display totalcost > sc.sol;
display x > sc.sol;
display _ampl_time >sc.sol;
display _total_solve_time >sc.sol;
exit;
```

5. 発電機起動停止問題

確率計画法に基づく起動停止問題モデルを以下の問題 (UC) に示す。台数 I の発電機による電力供給を考える。変数 u_{it} は発電機 i の時間 t における状態を表す 0-1 変数である。変数 x_{it} は発電機 i の時間 t の出力である。関数 $f(x_{it})$ は発電機 i の燃料費を表す x_{it} の凸 2 次関数である。関数 $g_i(u_{i,t-1}, u_{it})$ は発電機 i の起動費用を表し、 $(u_{i,t-1}, u_{it}) = (0, 1)$ の時に正の起動費用となり、それ以外の場合には 0 となる関数である。

$$(UC) : \min \sum_{t=1}^T g_i(u_{i,t-1}, u_{it}) + \sum_{i=1}^I \sum_{t=1}^T f_i(x_{it}) u_{it},$$

$$\text{subject to } \sum_{i=1}^I x_{it} \geq d_t, \forall t$$

$$u_{it} - u_{i,t-1} \leq u,$$

$$\tau = t+1, \dots, \min\{t+L_i - 1, T\}, \forall i, t=2, \dots, T$$

$$u_{i,t-1} - u_{it} \leq 1 - u_{it},$$

$$\tau = t+1, \dots, \min\{t+L_i - 1, T\}, \forall i, t=2, \dots, T$$

$$q_i u_{it} \leq x_{it} \leq Q_i u_{it}, \quad u_{it} \in \{0, 1\}, \forall i, \forall t,$$

目的関数は、供給コストの最小化である。供給コストは燃料費の全てのシナリオに対する期待値と起動費用の総和となる。第 1 制約は、出力の総和が電力需要を満たすための条件である。第 2 制約は、発電機 i は一旦起動したら L_i 時間連続で運転しなければならないことを表す。同様に第 3 制約は、発電機 i は一旦停止したら L_i 時間連続で停止しなければならないことを表す。第 4 制約は発電機の出力の上下限を与える。 Q_i, q_i はそれぞれ発電機 i の出力の上限値、下限値である。

#uc.mod

param N; #設備数

param T; #時間数

param a{i in 1..N};

param b{i in 1..N};

param c{i in 1..N};

param d{t in 1..T};

param g{i in 1..N};

param xmin{i in 1..N};

param xmax{i in 1..N};

param uptime{i in 1..N};

param downtime{i in 1..N};

var x{i in 1..N, t in 1..T} >=0;

var u{i in 1..N, t in 0..T} binary; #時間 t における状態を表す

var v{i in 1..N, t in 1..T} >=0; #時間 t に起動するかどうかを表す

```

minimize cost: sum{i in 1..N, t in 1..T} (a[i]*x[i,t]*x[i,t]+b[i]*x[i,t])
+sum{i in 1..N, t in 1..T} (c[i]*u[i,t])
+sum{i in 1..N, t in 1..T} g[i]*v[i,t];

subject to demand{t in 1..T}: sum{i in 1..N} x[i,t] >= d[t];
subject to minoutput{i in 1..N, t in 1..T}: xmin[i]*u[i,t] <= x[i,t];
subject to maxoutput{i in 1..N, t in 1..T}: x[i,t] <= xmax[i]*u[i,t];

subject to minup{i in 1..N, t in 2..T, s in t+1..min(t+uptime[i]-1,T)}:
    u[i,t]-u[i,t-1] <= u[i,s];
subject to mindown{i in 1..N, t in 2..T, s in t+1..min(t+downtime[i]-1,T)}:
    u[i,t-1]-u[i,t] <= 1- u[i,s];
subject to uv{i in 1..N, t in 1..T}: u[i,t]-u[i,t-1] <= v[i,t];
subject to init_u {i in 1..N}: u[i,0]=0;

```