Excelソルバー入門

『Excelソルバーではじめる最適化』 セッション2

堀田敬介

文教大学　経営学部

2024年2月3日（土）

本セミナーの構成

1. 数理最適化とソルバー（後藤）
2. Excel ソルバー入門（堀田）
3. 線形整数最適化（堀田）
4. ロバスト最適化（後藤）
5. VBA を使って便利にする（後藤）
6. データ包絡分析法(DEA)（後藤）

- Excelから次のステップへ（後藤）
- 閉会（閉会後　個別相談・質問コーナー）

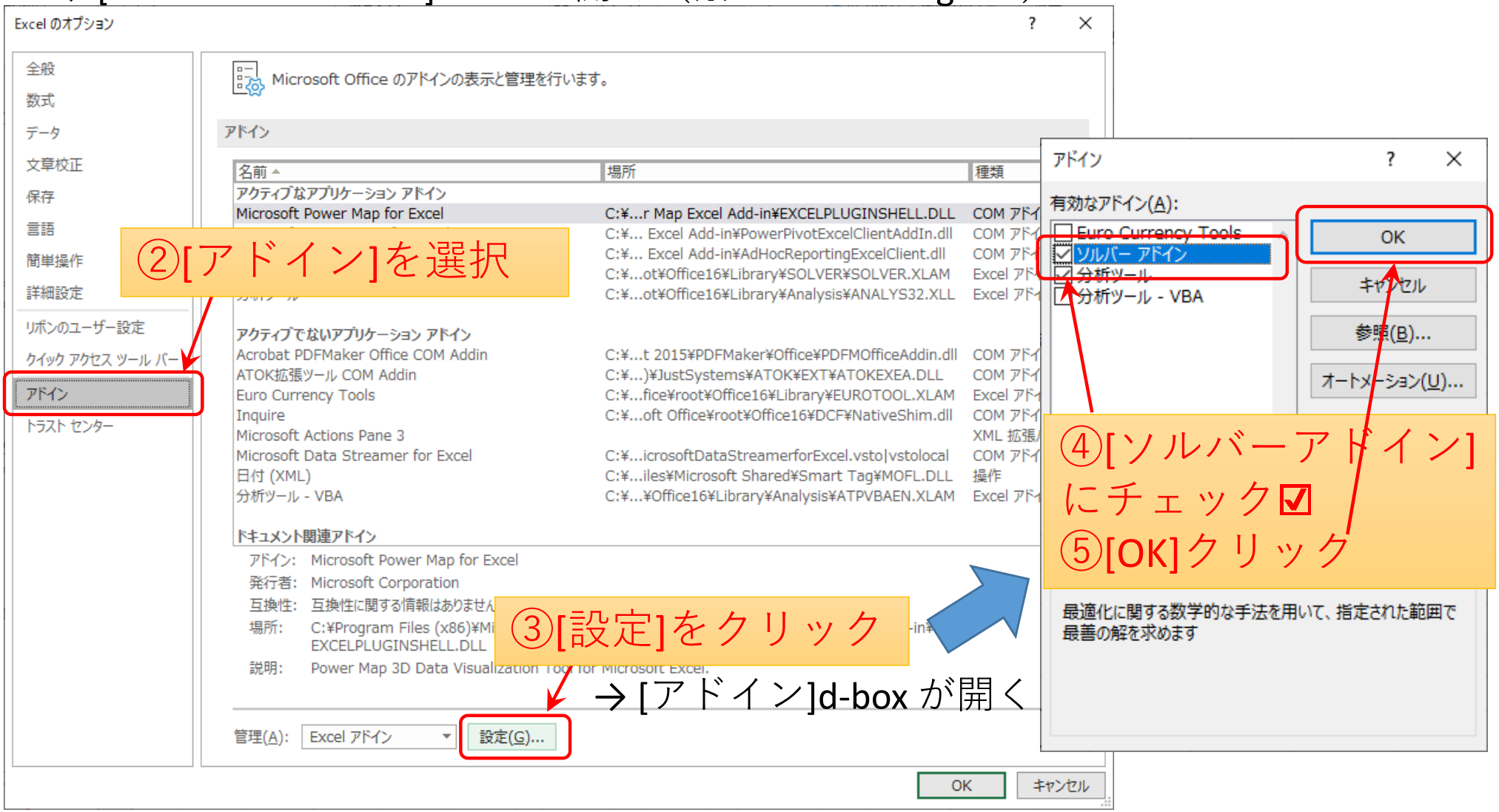
0. ソルバーの準備

Excelの初期状態では
ソルバーを使えない

- Excelソルバーを使える状態にする設定方法

① メニューから[ファイル]-[オプション]を選択

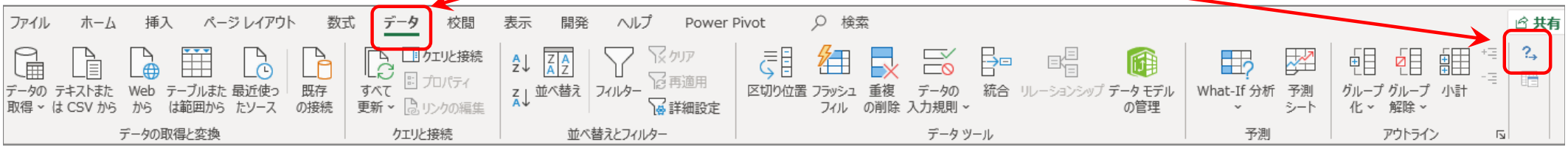
→ [Excelのオプション]d-box が開く （※d-box = dialog box）



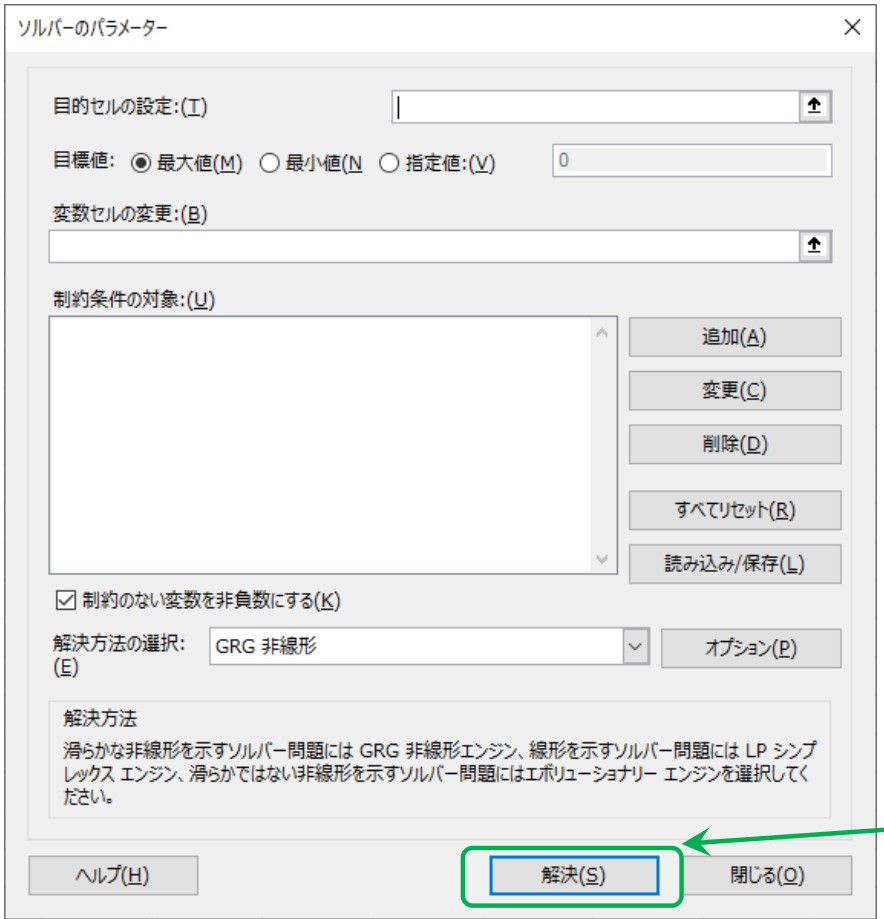
0. ソルバーの準備

- ソルバーの起動・設定・実行

起動 = メニューから[データ]-[ソルバー]を選択



→ [ソルバーのパラメーター] d-box が開く



目的関数の設定

変数(セル)の設定

制約条件の設定

手法の選択
オプション設定

実行(計算開始)

Outline

1. ともかく使ってみよう：LPを解く
2. ダイエット問題
3. 輸送問題
4. 生産計画
5. 生産スケジューリング
6. 非零和ゲームの均衡解とLP
7. 割当問題
8. クラス編成問題
9. 適当な問題を生成して解かせてみる

1. ともかく使ってみよう：LPを解く

- 線形最適化問題(Linear Optimization Problem; LP)

$$\begin{aligned}
\text{min.}\quad & 2x_1 + x_2 + 2x_3 + x_4 + 3x_5 \\
\text{s.t.}\quad & x_1 + 2x_3 + x_5 \geq 5 \\
& 9x_1 + 2x_2 + x_4 + 4x_5 \geq 1 \\
& x_2 + 5x_3 + x_5 \geq 3 \\
& x_1 + 3x_3 + x_5 \geq 2 \\
& x_1, x_2, x_3, x_4, x_5 \geq 0
\end{aligned}$$

- 上の定式化を行列・ベクトルで表記すると…

$$\begin{aligned}
& min.\, \boldsymbol{c}^T \boldsymbol{x} \\
& s.t.\, A\boldsymbol{x} \geq \boldsymbol{b} \\
& x \geq 0
\end{aligned}$$

$$\boldsymbol{c}^T = (2 \quad 1 \quad 2 \quad 1 \quad 3)$$

$$A = \begin{pmatrix} 1 & 0 & 2 & 0 & 1 \\ 9 & 2 & 0 & 1 & 4 \\ 0 & 1 & 5 & 0 & 1 \\ 1 & 0 & 3 & 0 & 1 \end{pmatrix}, \boldsymbol{b} = \begin{pmatrix} 5 \\ 1 \\ 3 \\ 2 \end{pmatrix} \quad \boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix}$$

1. ともかく使ってみよう：LPを解く

- 線形最適化問題をExcelシートに記述

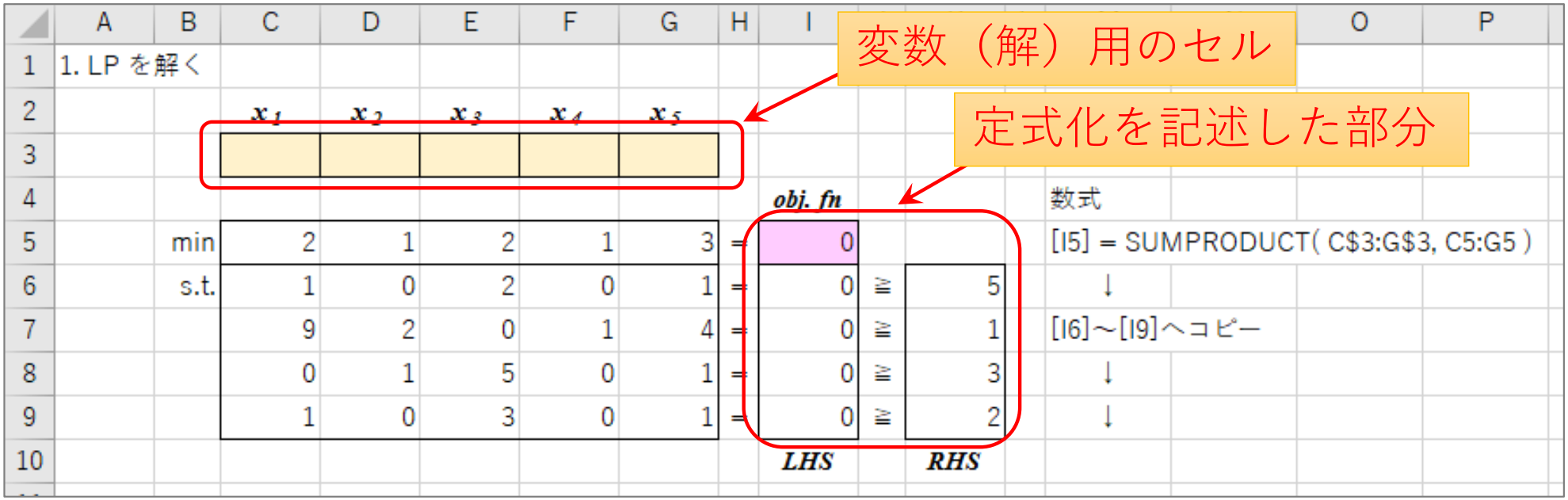
$$
\begin{aligned}
\text{min.}\quad & 2x_1 + x_2 + 2x_3 + x_4 + 3x_5 \\
\text{s.t.}\quad & x_1 + 2x_3 + x_5 \geq 5 \\
& 9x_1 + 2x_2 + x_4 + 4x_5 \geq 1 \\
& x_2 + 5x_3 + x_5 \geq 3 \\
& x_1 + 3x_3 + x_5 \geq 2 \\
& x_1, x_2, x_3, x_4, x_5 \geq 0
\end{aligned}
$$

行列・ベクトルによる定式化の表記

$$
min.\, \boldsymbol{c}^T \boldsymbol{x} \qquad s.t.\, A\boldsymbol{x} \geq \boldsymbol{b} \qquad \boldsymbol{x} \geq 0
$$

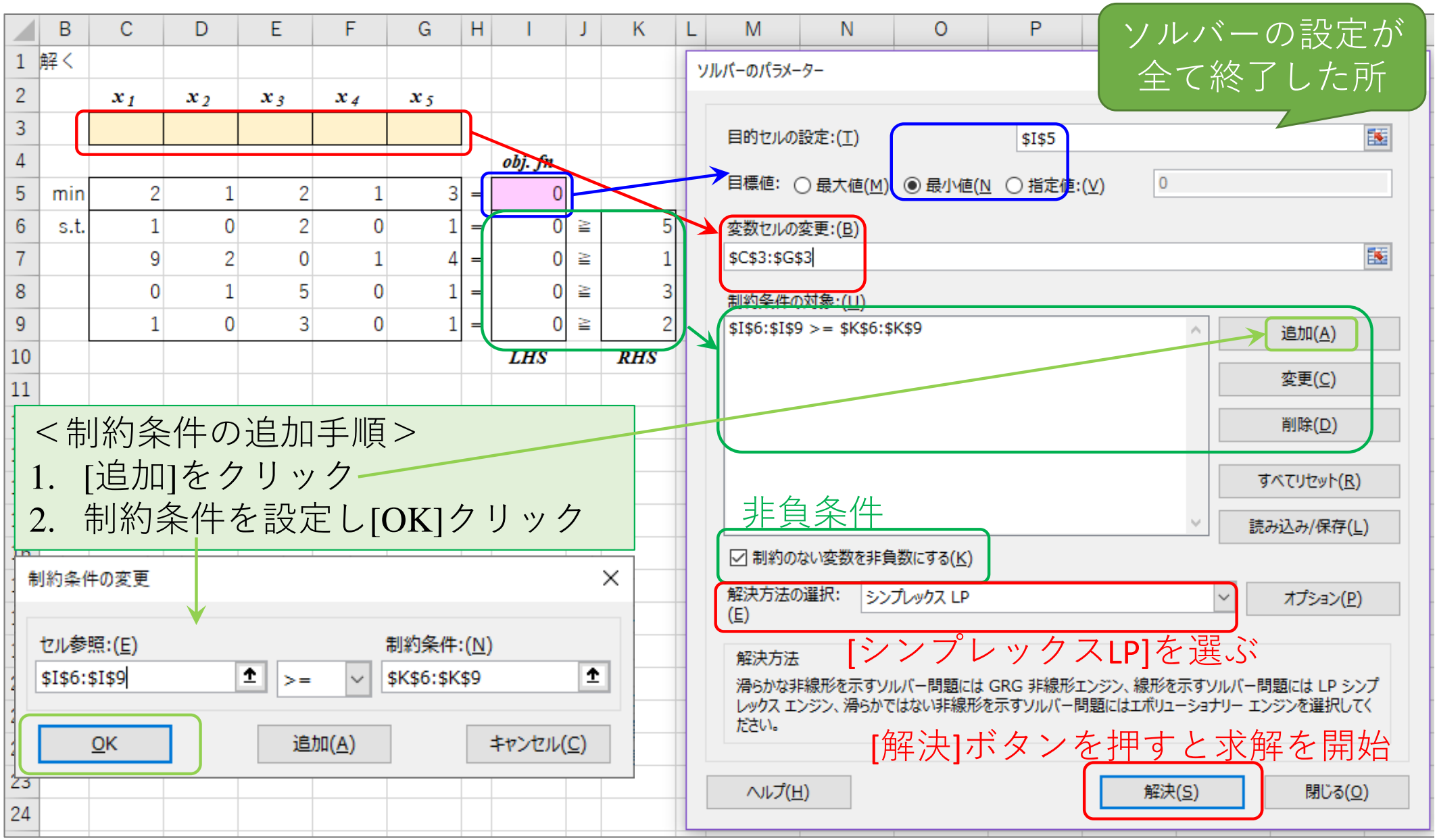
$$
\boldsymbol{c}^T = (2 \quad 1 \quad 2 \quad 1 \quad 3)
$$

$$
A = \begin{pmatrix} 1 & 0 & 2 & 0 & 1 \\ 9 & 2 & 0 & 1 & 4 \\ 0 & 1 & 5 & 0 & 1 \\ 1 & 0 & 3 & 0 & 1 \end{pmatrix}, \boldsymbol{b} = \begin{pmatrix} 5 \\ 1 \\ 3 \\ 2 \end{pmatrix}
$$



1. ともかく使ってみよう：LPを解く

- Excelシート上の内容をソルバーへ設定する



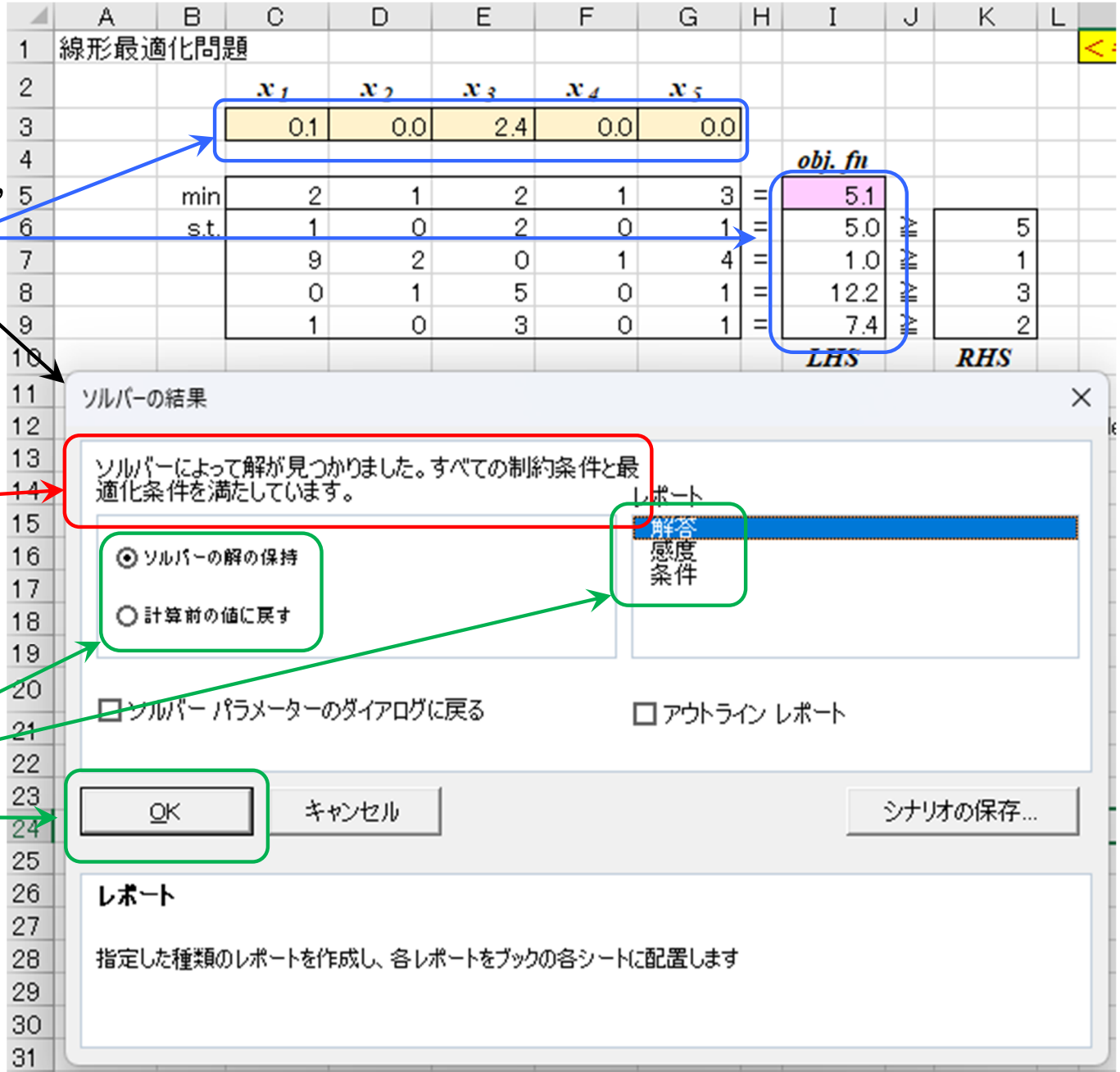
1. ともかく使ってみよう：LPを解く

- 結果がExcelシート上に反映される

求解が終わると，
[ソルバーの結果]d-box が開き，
シート上に結果が反映される
（※前頁と比較せよ）

最適解(Optimal Solution)
が見つかった場合の
メッセージ

オプションを設定して
[OK]で終了



1. ともかく使ってみよう：LPを解く

【演習】

- 以下のLPについてExcel solverで最適解を求めよ

$$\begin{aligned} \text{min.} \quad & x_1 + x_2 \\ \text{s.t.} \quad & -4x_1 + 3x_2 \leq 12 \\ & x_1 - 2x_2 \leq 2 \\ & 2x_1 + x_2 \leq 4 \\ & -2x_1 - x_2 \leq 6 \\ & x_1, x_2 \geq 0 \end{aligned}$$

$$\begin{aligned} & min.\ \boldsymbol{c}^T\boldsymbol{x} \\ & s.t.\ A\boldsymbol{x} \leq \boldsymbol{b} \\ & \boldsymbol{x} \geq 0 \end{aligned} \qquad \boldsymbol{c}^T = (1 \quad 1)$$

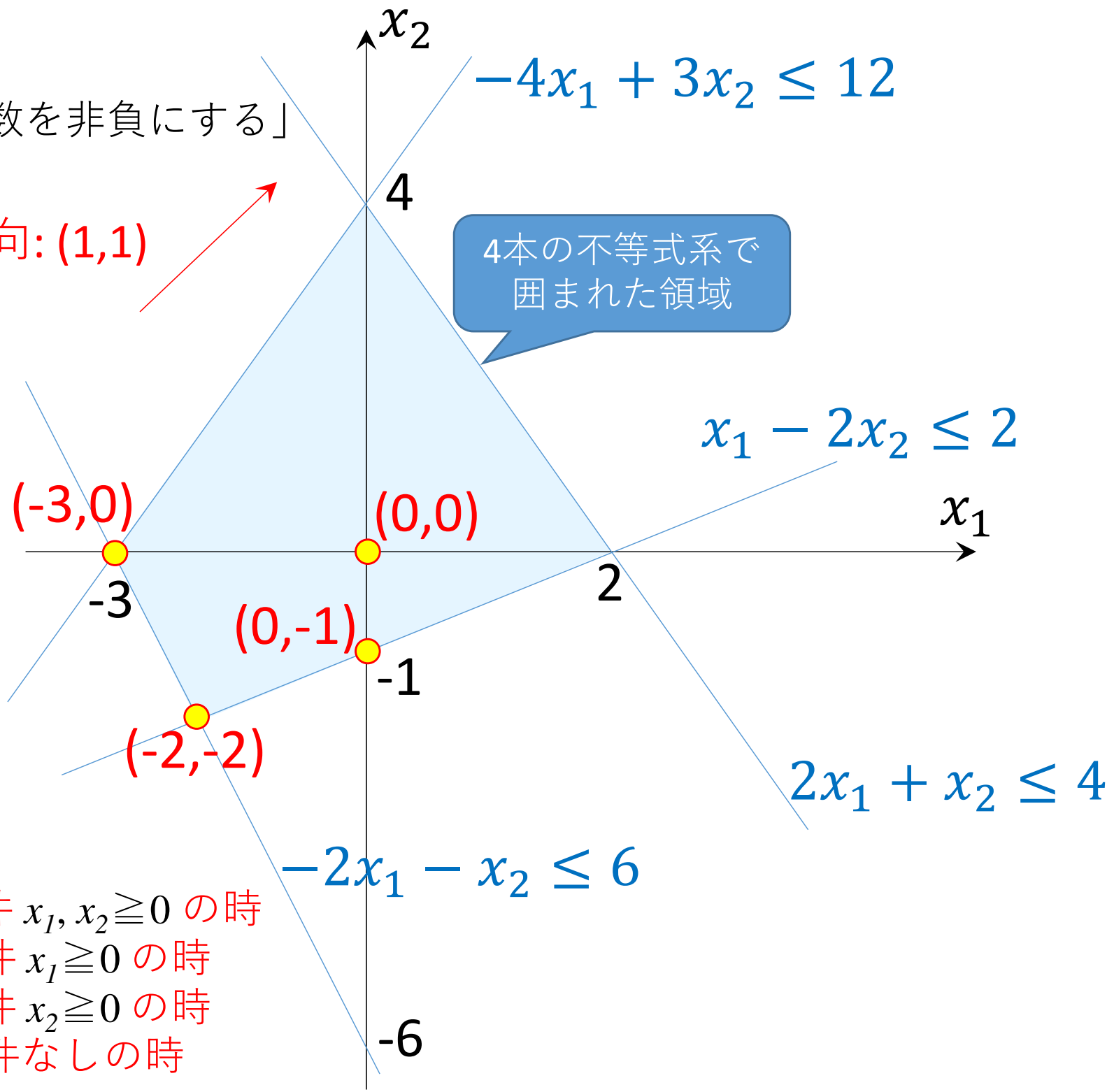
$$A = \begin{pmatrix} -4 & 3 \\ 1 & -2 \\ 2 & 1 \\ -2 & -1 \end{pmatrix}, \boldsymbol{b} = \begin{pmatrix} 12 \\ 2 \\ 4 \\ 6 \end{pmatrix}$$

	A	B	C	D	E	F	G	H	I
1	「制約のない変数を非負にする」へのチェック								
2			x_1	x_2					
3									
4									
5		min	1	1	=	0			
6		s.t.	-4	3	=	0	≦	12	
7			1	-2	=	0	≦	2	
8			2	1	=	0	≦	4	
9			-2	-1	=	0	≦	6	
10						*LHS*		*RHS*	
11									
12		[F5]	= SUMPRODUCT(C$3:D$3, C5:D5)						
13			→[F5]をコピーし，[F6:F9]へ貼り付け						

【補足】

「☑制約のない変数を非負にする」の動作確認

目的関数の方向: (1,1)
目的：最小化



【最適解】

- ✓ (0, 0) … 非負条件 $x_1, x_2 \geqq 0$ の時
- ✓ (0,-1) … 非負条件 $x_1 \geqq 0$ の時
- ✓ (-3, 0) … 非負条件 $x_2 \geqq 0$ の時
- ✓ (-2,-2) … 非負条件なしの時

【補足】

「☑制約のない変数を非負にする」の動作確認

✓ 4つのケースで最適解がどう変わるか（どの制約が活きるか）確認

① ☑制約のない変数を非負にする

② ☐制約のない変数を非負にする

③ ☑制約のない変数を非負にする & 冗長制約「$x_1 \geqq -4$」を追加

④ ☑制約のない変数を非負にする & 冗長制約「$x_2 \geqq -4$」を追加

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	「制約のない変数を非負にする」へのチェックについて動作確認												opt. sol.			つまり追加される制約は			
2			x_1	x_2									x_1	x_2		x_1	x_2		
3										☑	制約のない変数を非負にする	→	0	0		$x_1 \geqq 0$	$x_2 \geqq 0$		
4																			
5		min	1	1	=	0				☐	制約のない変数を非負にする	→	−2	−2		なし	なし		
6		s.t.	−4	3	=	0	≦	12											
7			1	−2	=	0	≦	2		☑	制約のない変数を非負にする	→	−3	0		なし	$x_2 \geqq 0$		
8			2	1	=	0	≦	4			& 冗長制約[$x_1 \geqq -4$]								
9			−2	−1	=	0	≦	6			※ x_1に制約を追加した．つまり「x_1は制約のない変数」ではなくなるので非負条件がつかなくなる								
10						*LHS*		*RHS*		☑	制約のない変数を非負にする	→	0	−1		$x_1 \geqq 0$	なし		
11											& 冗長制約[$x_2 \geqq -4$]								
12		[F5]	= SUMPRODUCT(C$3:D$3, C5:D5)								※ x_2に制約を追加した．つまり「x_2は制約のない変数」ではなくなるので非負条件がつかなくなる								
13			→[F5]をコピーし，[F6:F9]へ貼り付け																

想定外のミスをなくすために，明瞭な場合を除いては「☑制約のない変数を非負にする」のチェックは外し，自分で**陽に非負条件の有無を設定**した方が良い

【参考】Python-MIP で解く

- Google Colaboratory を開く
  - 利用方法（初回）

    (1) google アカウントにログインし，google drive へ移動

    (2)「新規」ー「その他」ー「アプリを追加」を選択

    (3)「Google Colaboratory」を追加
  - 利用方法（2回目以降）

    (1) google アカウントにログインし，google drive へ移動

    (2)「新規」ー「その他」ー「Google Colaboratory」を選択

  - ファイル名は default では [Untitled0.ipynb] となっている．変更可．

  - ファイルは google drive に自動保存される．一度作成したら，次回以降は，google drive 内のファイル [***.ipynb] を選択して，開くことができる

※この拡張子名は IPython Notebook の略で，Jupyter Notebook 専用のファイルということ．IPython は Python を対話的に実行する環境の1つで，Jupyter Notebook とは，それをブラウザ上で動かす実行環境

【参考】Python-MIP で解く

- Python-mip インストール

[コード]の欄にこのように記入
左の三角ボタンを押して「実行」
（このコードの下に，メッセージが
表示されるので終わるまで待つ）

- 線形最適化問題の記述1（係数の設定）
  - 左上の[＋コード]ボタンを押して，次の記述欄（[コード]欄）を追加する
  - 以下の通りに記述し，左の三角ボタンを押して「実行」する

```
c = [2, 1, 2, 1, 3]
b = [5, 1, 3, 2]
A = [[1, 0, 2, 0, 1],
     [9, 2, 0, 1, 4],
     [0, 1, 5, 0, 1],
     [1, 0, 3, 0, 1]]
J = range(len(c))
I = range(len(b))
```

例題の線形最適化問題について
c = […] : 目的関数の係数ベクトル
b = […] : 制約条件の右辺ベクトル
A = […] : 制約条件の左辺係数行列
J = range(len(c)) : 列の添え字の範囲
I = range(len(b)) : 行の添え字の範囲
※ len(…) は … のサイズlengthを返す関数
を設定しているところ

【参考】Python-MIP で解く

- 線形最適化問題の記述2（定式化と求解）
  - 左上の [＋コード] をクリックして，以下のコードを記述する欄を追加する
  - そこに以下の通りにコードを記述した後，左三角ボタンで実行する

※ ＃より右はコメント
（プログラムとは関係ない，人間用の記述）

定式化

```
from mip.model import *

m = Model("lpex1")  # モデルの設定

x = [m.add_var(var_type="C", lb=0) for j in J]  # 変数宣言：モデル m に変数を追加
m.objective = minimize(xsum(c[j] * x[j] for j in J))  # 目的関数の設定：モデル m に目的関数を追加
for i in I:
  m += xsum(A[i][j] * x[j] for j in J) >= b[i]       # 制約条件の設定：モデル m に制約条件を追加

m.optimize()   # 最適化（求解）の実行

if m.status.value==0:   # もし，最適解が求まったなら
  print("最適解:")      # 最適解を表示
  for j in J:
    print(" x[",j,"] = ", x[j].x)
  print("最適値:", m.objective_value, "=", m.objective) # 目的関数値を表示
else:                   # もし，最適解が求まらなかったなら
  print("error:最適解は求まりませんでした")  # エラーメッセージを表示
```

最適解
と
最適値
の表示

※"C" = continuous（連続）
連続変数とするということ
※ lb = lower bound（下限）
非負条件に該当する．
default は lb=0 なので，記述しなくても可

```
最適解:
 x[ 0 ] =  0.1111111111111111
 x[ 1 ] =  0.0
 x[ 2 ] =  2.4444444444444446
 x[ 3 ] =  0.0
 x[ 4 ] =  0.0
最適値: 5.111111111111112 = + 2.0var(0) + var(1) + 2.0var(2) + var(3) + 3.0var(4)
```

実行すると，結果を表示

2. ダイエット問題

- 例題

『あだち王国』には，4つの食べ物「神秘ケーキ」「魅惑菓子」「苦渋野菜」「過酸果物」がある．各食べ物は，3つの栄養素「だんはっく」「ガルジウム」「ヒタビン」と，3つの食品含有物「糖分」「塩分」「ガロリー」が含まれている．食べ物1単位当たりの含有量は表の通りである

	食べ物	神秘ケーキ	魅惑菓子	苦渋野菜	過酸果物
栄養素	だんはっく	3	1	4	2
	ガルジウム	1	2	2	1
	ヒタビン	2	1	2	5
含有物	糖分	7	5	3	4
	塩分	1	2	4	8
	ガロリー	40	50	55	20

『あだち王国』人は，3つの栄養素を一日に各々最低50, 40, 60は摂取しないと死ぬ！　また，糖分と塩分は各々一日に150を超えると過剰摂取で死ぬ！！

死なないようにしながら，ガロリーを最小にする食べ物の量を知りたい

【変数設定】4つの食べ物を食べる量を x_1, x_2, x_3, x_4 とする

2. ダイエット問題

- 例題：定式化

	食べ物	神秘ケーキ	魅惑菓子	苦渋野菜	過酸果物
栄養素	だんはっく	3	1	4	2
	ガルジウム	1	2	2	1
	ヒタビン	2	1	2	5
含有物	糖分	7	5	3	4
	塩分	1	2	4	8
	ガロリー	40	50	55	20

$$\text{min.} \quad 40x_1 + 50x_2 + 55x_3 + 20x_4$$ ← ガロリー最小化

$$\text{s.t.} \quad 3x_1 + x_2 + 4x_3 + 2x_4 \geq 50$$
$$x_1 + 2x_2 + 2x_3 + x_4 \geq 40$$
$$2x_1 + x_2 + 2x_3 + 5x_4 \geq 60$$

3栄養素の必要量摂取条件

$$7x_1 + 5x_2 + 3x_3 + 4x_4 \leq 150$$
$$x_1 + 2x_2 + 4x_3 + 8x_4 \leq 150$$

2食品含有物の過剰量摂取条件

$$x_1, x_2, x_3, x_4 \geq 0$$ ← 食品摂取量は非負

$$min.\, c^T x \quad s.t.\, Ax \geq b \quad x \geq 0$$

$$c^T = (40 \quad 50 \quad 55 \quad 20)$$

$$A = \begin{pmatrix} 3 & 1 & 4 & 2 \\ 1 & 2 & 2 & 1 \\ 2 & 1 & 2 & 5 \\ -7 & -5 & -3 & -4 \\ -1 & -2 & -4 & -8 \end{pmatrix}, b = \begin{pmatrix} 50 \\ 40 \\ 60 \\ -150 \\ -150 \end{pmatrix}$$

2. ダイエット問題

- 例題：Excel記述&ソルバー設定

	A	B	C	D	E	F	G	H	I	J
1	2. ダイエット問題									
2										
3		栄養素	神秘ケーキ	魅惑菓子	苦渋野菜	過酸果物		摂取量		必要量
4		だんはっく	3	1	4	2		0	≧	50
5		ガルジウム	1	2	2	1		0	≧	40
6		ヒタピン	2	1	2	5		0	≧	60
7		糖分	-7	-5	-3	-4		0	≧	-150
8		塩分	-1	-2	-4	-8		0	≧	-150
9		ガロリー	40	50	55	20		0		過剰量
10										
11			x_1	x_2	x_3	x_4				
12		食事量								
13										
14		[H4]	= SUMPRODUCT(C4:F4, C$12:F$12)							
15			→[H4]をコピーし，[H5:H9]へ貼り付け							

ソルバーの設定が全て終わったところ

2. ダイエット問題

- 例題：結果

	A	B	C	D	E	F	G	H	I	J	K
1		ダイエット問題									
2											
3		栄養素	神秘ケーキ	魅惑菓子	苦渋野菜	過酸果物		摂取量		必要量	
4		だんはっく	3	1	4	2		50	≧	50	
5		ガルジウム	1	2	2	1		40	≧	40	
6		ヒタビン	2	1	2	5		90	≧	60	
7		糖分	-7	-5	-3	-4		-117.5	≧	-150	
8		塩分	-1	-2	-4	-8		-150	≧	-150	
9		ガロリー	40	50	55	20		937.5		過剰量	
10											
11			x_1	x_2	x_3	x_4					
12		食事量	0	10	2.5	15					
13											

【参考】Python-MIP で解く

- 記述1：係数設定

```
c = [40, 50, 55, 20]
b = [50, 40, 60, -150, -150]
A = [[3, 1, 4, 2],
     [1, 2, 2, 1],
     [2, 1, 2, 5],
    [-7,-5,-3,-4],
    [-1,-2,-4,-8]]
I, J = range(len(b)),range(len(c))
```

- 記述2：定式化と求解

定式化

```
from mip.model import *

m = Model("DietEx1")  # モデルの設定

x = [m.add_var(var_type="C", lb=0) for j in J]        # 変数宣言：モデル m
m.objective = minimize(xsum(c[j] * x[j] for j in J))  # 目的関数の設定：モ
for i in I:
  m += xsum(A[i][j] * x[j] for j in J) >= b[i]        # 制約条件の設定：モ

m.optimize()   # 最適化（求解）の実行
```

最適解と最適値の表示

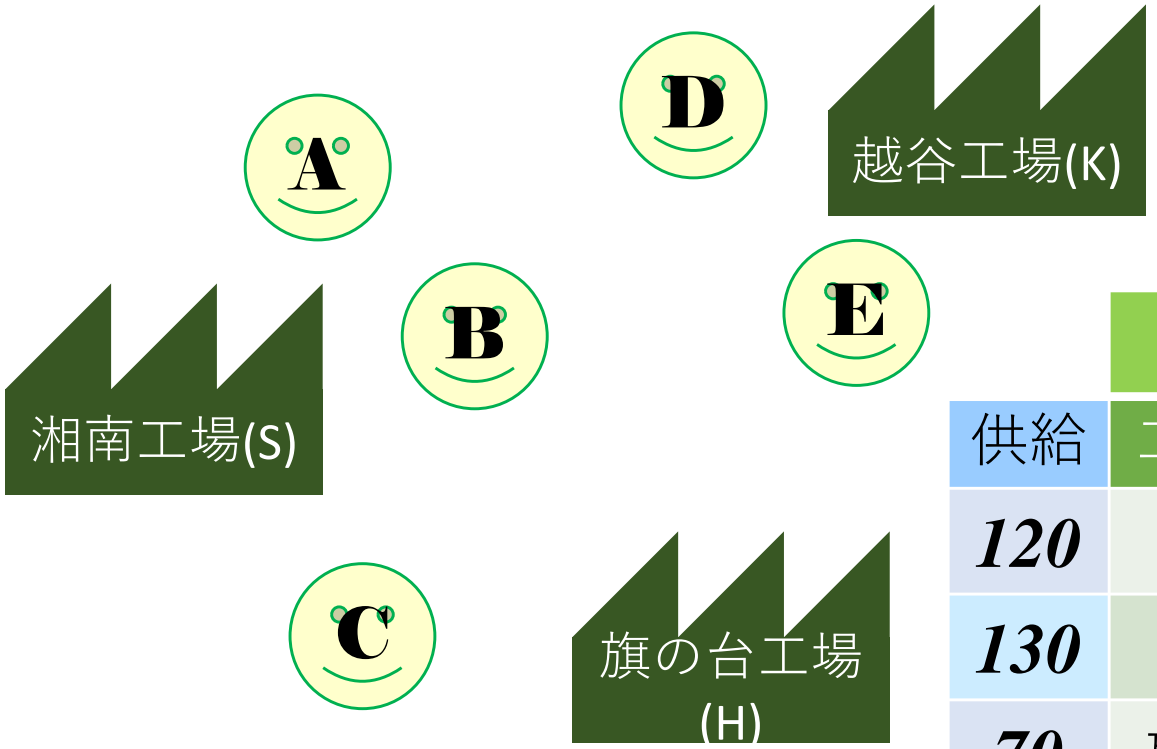
```
if m.status.value==0:   # もし，最適解が求まったなら
  print("最適解:")     # 最適解を表示
  for j in J:
    print(" x[",j,"] = ", x[j].x)
  print("最適値:", m.objective_value, "=", m.objective) # 目的関数値を表示
else:                   # もし，最適解が求まらなかったなら
  print("error:最適解は求まりませんでした")  # エラーメッセージを表示
```

```
最適解:
 x[ 0 ] =  0.0
 x[ 1 ] =  9.999999999999998
 x[ 2 ] =  2.5000000000000013
 x[ 3 ] =  14.999999999999998
最適値: 937.5 = + 40.0var(0) + 50.0var(1) + 55.0var(2) + 20.0var(3)
```

3. 輸送問題

- 例題

文教重工は3つの工場（湘南・越谷・旗の台）があり，それぞれ製品を供給できる
顧客は5人いて，それぞれ需要（製品を欲しい量）がある
3つの工場から5人の顧客それぞれへの単位あたり輸送コストは表の通り
輸送コストが最小となる配送計画をたてたい



- ✓ 工場の供給量
- ✓ 顧客の需要量
- ✓ 工場から顧客へ製品を1単位

配送するのにかかる輸送コスト表

	需要	50	80	60	70	40
供給	工場＼顧客	A	B	C	D	E
120	湘南(S)	3	2	4	5	8
130	越谷(K)	5	6	5	3	2
70	旗の台(H)	7	3	1	2	3

【変数設定】工場 i →顧客 j への輸送量を変数 x_{ij} とする

$i \in \{S,K,H\}$
$j \in \{A,B,C,D,E\}$

3. 輸送問題

- 例題：定式化

	需要	50	80	60	70	40
供給	工場＼顧客	A	B	C	D	E
120	湘南(S)	3	2	4	5	8
130	越谷(K)	5	6	5	3	2
70	旗の台(H)	7	3	1	2	3

$$\begin{aligned}
\text{min.}\quad & 3x_{SA}+2x_{SB}+4x_{SC}+5x_{SD}+8x_{SE} \\
& +5x_{KA}+6x_{KB}+5x_{KC}+3x_{KD}+2x_{KE} \\
& +7x_{HA}+3x_{HB}+x_{HC}+2x_{HD}+3x_{HE} \\
\text{s.t.}\quad & x_{SA}+x_{SB}+x_{SC}+x_{SD}+x_{SE}\le 120 \\
& x_{KA}+x_{KB}+x_{KC}+x_{KD}+x_{KE}\le 130 \\
& x_{HA}+x_{HB}+x_{HC}+x_{HD}+x_{HE}\le 70 \\
& x_{SA}+x_{KA}+x_{HA}=50 \\
& x_{SB}+x_{KB}+x_{HB}=80 \\
& x_{SC}+x_{KC}+x_{HC}=60 \\
& x_{SD}+x_{KD}+x_{HD}=70 \\
& x_{SE}+x_{KE}+x_{HE}=40 \\
& x_{SA},\dots,x_{HE}\ge 0
\end{aligned}$$

輸送コスト最小化

工場の供給量条件

顧客の需要量条件

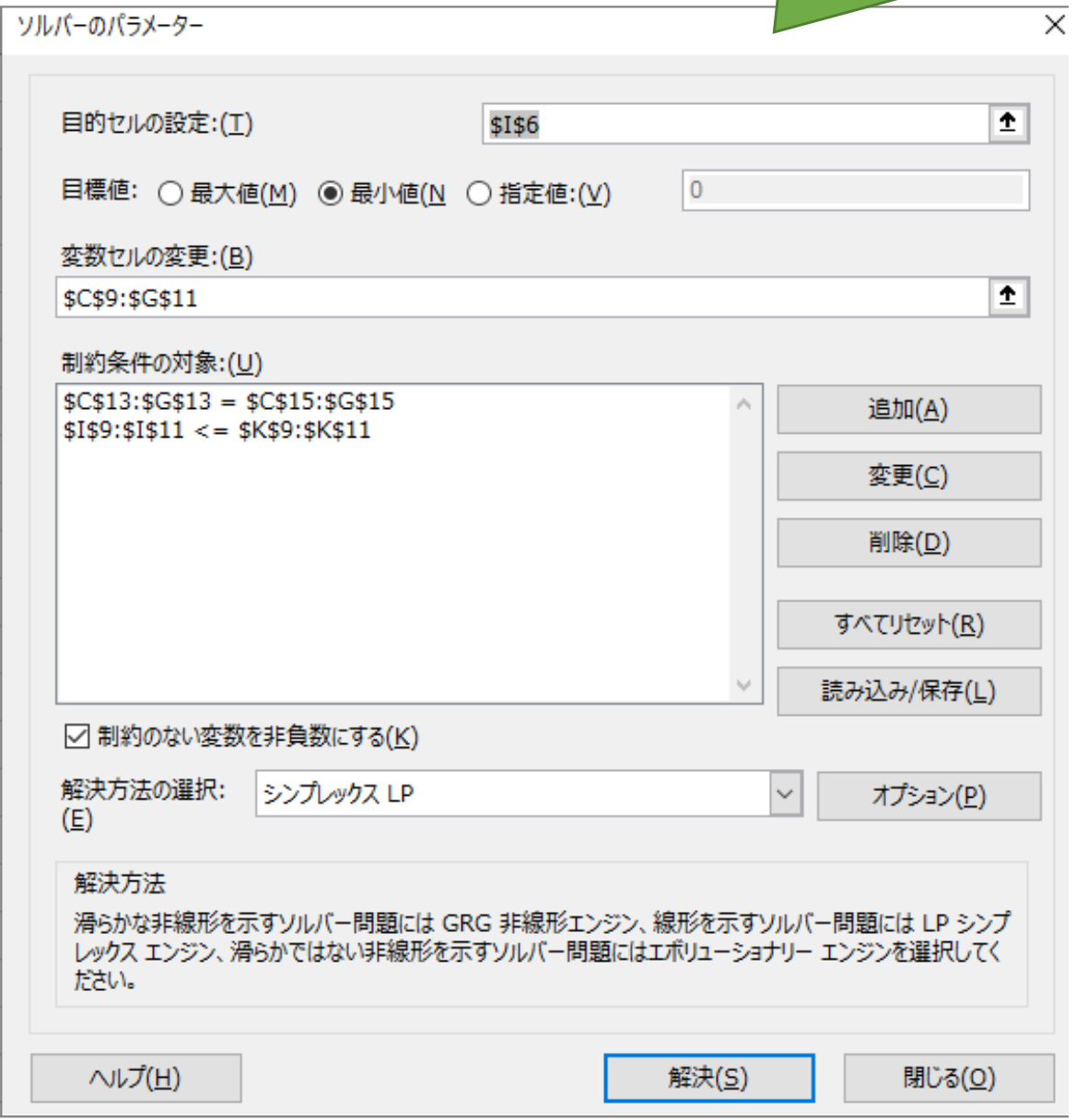
← 輸送量は非負

3. 輸送問題

- 例題：Excel記述&ソルバー設定

ソルバーの設定が全て終わったところ

	A	B	C	D	E	F	G	H	I	J	K
1	3. 輸送問題										
2		輸送コスト									
3		工場＼顧客	A	B	C	D	E				
4		湘南(S)	3	2	4	5	8				
5		越谷(K)	5	6	5	3	2		総コスト		
6		旗の台(H)	7	3	1	2	3		0		
7											
8		x_{ij}	A	B	C	D	E		輸送量		供給
9		S							0	≦	120
10		K							0	≦	130
11		H							0	≦	70
12											
13		輸送量	0	0	0	0	0				
14			=	=	=	=	=				
15		需要	50	80	60	70	40				
16											
17			[I9] = SUM(C9:G9)								
18			→[I9]をコピーし，[I10:I11]へ貼り付け								
19			[C13] = SUM(C9:C11)								
20			→[C13]をコピーし，[D13:G13]へ貼り付け								
21			[I6] = SUMPRODUCT(C4:G6, C9:G11)								
22											



3. 輸送問題

- 例題：結果

	A	B	C	D	E	F	G	H	I	J	K
1	輸送問題										
2		輸送コスト									
3		工場＼顧客	A	B	C	D	E				
4		湘南(S)	3	2	4	5	8				
5		越谷(K)	5	6	5	3	2		総コスト		
6		旗の台(H)	7	3	1	2	3		670		
7											
8		x_{ij}	A	B	C	D	E		輸送量		供給
9		S	40	80	0	0	0		120	≦	120
10		K	10	0	0	60	40		110	≦	130
11		H	0	0	60	10	0		70	≦	70
12											
13		輸送量	50	80	60	70	40				
14			=	=	=	=	=				
15		需要	50	80	60	70	40				
16											

【参考】Python-MIPで解く

- 係数設定

```
d = [50, 80, 60, 70, 40]  # 需要
s = [120, 130, 70]        # 供給
C = [[3, 2, 4, 5, 8],   # 輸送コスト
     [5, 6, 5, 3, 2],
     [7, 3, 1, 2, 3]]
J = range(len(d))
I = range(len(s))
```

- 定式化
- 求解

定式化

```
from mip.model import *

m = Model("trex1")  # モデルの設定（輸送問題）

x = [[m.add_var(var_type="C", lb=0) for j in J] for i in I]            # 変数宣言：輸送量
m.objective = minimize(xsum(C[i][j] * x[i][j] for i in I for j in J))  # 目的関数：輸送コスト最小化
for j in J:
  m += xsum(x[i][j] for i in I) == d[j]  # 制約条件1：需要
for i in I:
  m += xsum(x[i][j] for j in J) <= s[i]  # 制約条件2：供給

m.optimize()   # 最適化（求解）の実行

if m.status.value==0:   # もし，最適解が求まったなら
  print("最適解: ")     # 最適解を表示
  for i in I:
    for j in J:
      print(" x[",i,",",j,"] = ", x[i][j].x)

  print(" 目的関数値: ", m.objective_value, "=", m.objective) # 目的関数値を表示
else:                   # もし，最適解が求まらなかったなら
  print(" error: 最適解は求まりませんでした")  # エラーメッセージを表示
```

＜結果＞

最適解と最適値の表示

```
最適解:
 x[ 0 , 0 ] =  40.0
 x[ 0 , 1 ] =  80.0
 x[ 0 , 2 ] =  0.0
 x[ 0 , 3 ] =  0.0
 x[ 0 , 4 ] =  0.0
 x[ 1 , 0 ] =  10.0
 x[ 1 , 1 ] =  0.0
 x[ 1 , 2 ] =  0.0
 x[ 1 , 3 ] =  60.0
 x[ 1 , 4 ] =  40.0
 x[ 2 , 0 ] =  0.0
 x[ 2 , 1 ] =  0.0
 x[ 2 , 2 ] =  60.0
 x[ 2 , 3 ] =  10.0
 x[ 2 , 4 ] =  0.0
 目的関数値:  670.0 = + 3.0var(0) + 2.0var(1) + 4.0var(2) + 5.0var(3) + 8.0var(4) + 5.0var(5) + 6.0var(6) + 5.0var(7) + 3.0var(8) + 2.0var(9) + 7.
```

4. 生産計画

- 例題

文教重工は3つの材料（P,Q,R）を使い，5つの液体製品（A,B,C,D,E）を作っている
製品1単位作るのに必要な変動費，各材料の量，製造時間はそれぞれ表の通り
また，販売先への契約上の販売量下限および上限と，販売価格も表の通り
共通の固定費は¥5,000,000，工場の総稼働時間は14,000(h)，所持材料（P,Q,R）の量は
それぞれ4,900 / 5,200 / 4,200 である．利益が最大になる各製品の生産量を求めたい

製品	A	B	C	D	E
変動費	¥500	¥550	¥570	¥490	¥400
材料P	1.1	0.0	0.0	1.2	0.9
材料Q	0.0	0.8	0.9	0.1	0.0
材料R	0.1	0.6	0.4	0.0	0.7
製造時間(h)	0.75	1.10	1.20	1.50	2.00
最低販売量	2500	4000	2000	1200	700
最高販売量	5000	5500	4000	3000	3500
販売価格	¥2,000	¥3,000	¥3,500	¥4,000	¥4,500

【変数設定】 製品 i の生産量を変数 x_i とする　　$i \in \{A,B,C,D,E\}$

4. 生産計画

- 例題：定式化

※「固定費」は定数なので，
目的関数式に入れなくてもよい

利益（＝販売額－変動費－固定費）の最大化

max. $2000x_A + 3000x_B + 3500x_C + 4000x_D + 4500x_E$
$-500x_A - 550x_B - 570x_C - 490x_D - 400x_E - 5000000$

s.t.

$$1.1x_A + 0.0x_B + 0.0x_C + 1.2x_D + 0.9x_E \leq 4900$$
$$0.0x_A + 0.8x_B + 0.9x_C + 0.1x_D + 0.0x_E \leq 5200$$
$$0.1x_A + 0.6x_B + 0.4x_C + 0.0x_D + 0.7x_E \leq 4200$$

材料条件

$$0.75x_A + 1.1x_B + 1.2x_C + 1.5x_D + 2.0x_E \leq 14000$$

時間条件

$$2500 \leq x_A \leq 5000, \quad 4000 \leq x_B \leq 5500,$$
$$2000 \leq x_C \leq 4000, \quad 1200 \leq x_D \leq 3000,$$
$$700 \leq x_E \leq 3500$$

契約条件

$$x_A, \dots, x_E \geq 0$$

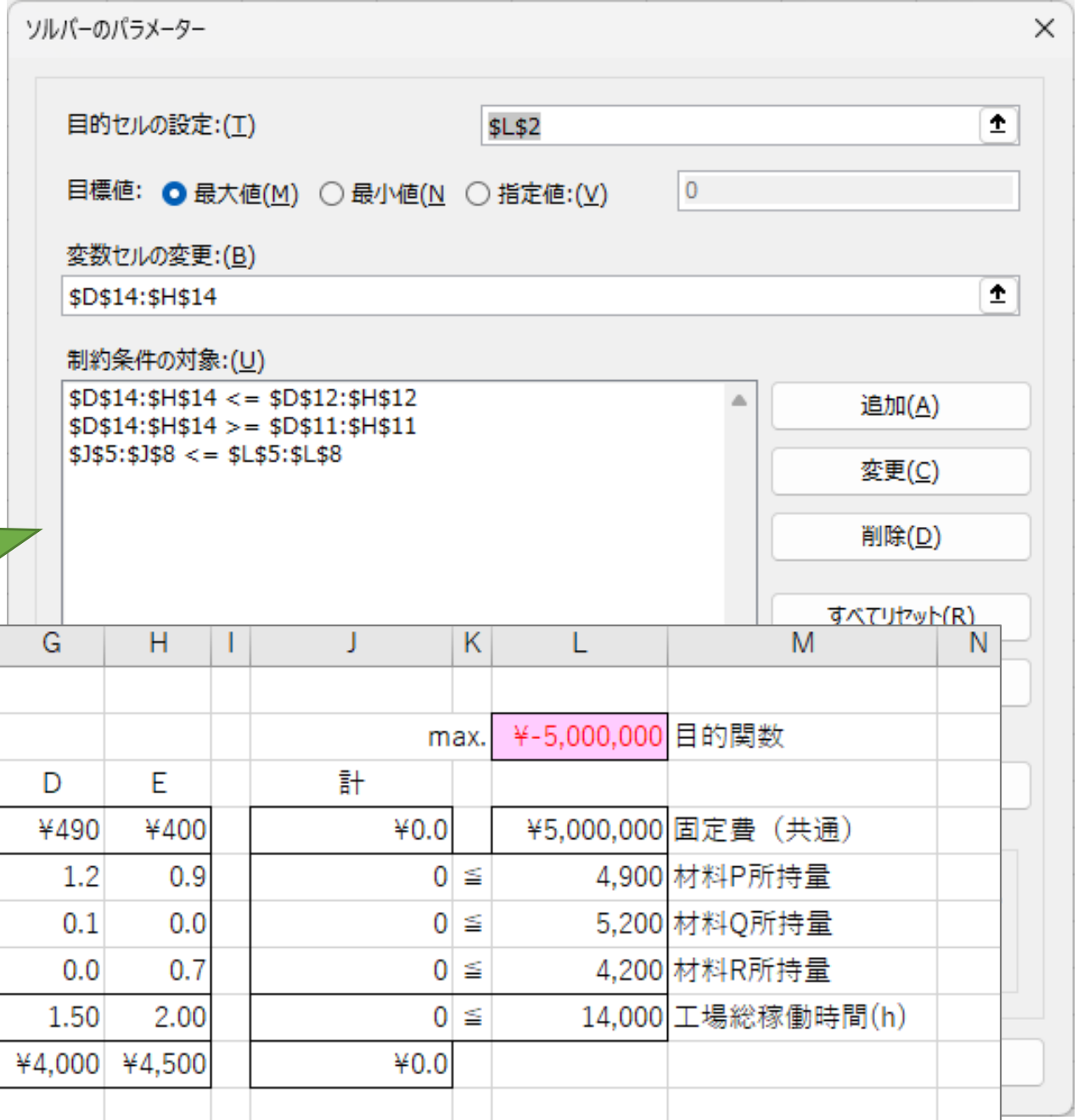
← 生産量は非負

※全変数に下限制約があるので，非負条件はなくてもよい

4. 生産計画

- 例題：Excel記述
  &ソルバー設定

ソルバーの設定が全て終わったところ



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	4. 生産計画													
2			1単位あたり							max.		¥-5,000,000	目的関数	
3			製品	A	B	C	D	E		計				
4			変動費	¥500	¥550	¥570	¥490	¥400		¥0.0		¥5,000,000	固定費（共通）	
5			材料P	1.1	0.0	0.0	1.2	0.9		0	≦	4,900	材料P所持量	
6			材料Q	0.0	0.8	0.9	0.1	0.0		0	≦	5,200	材料Q所持量	
7			材料R	0.1	0.6	0.4	0.0	0.7		0	≦	4,200	材料R所持量	
8			製造時間(h)	0.75	1.10	1.20	1.50	2.00		0	≦	14,000	工場総稼働時間(h)	
9			販売価格	¥2,000	¥3,000	¥3,500	¥4,000	¥4,500		¥0.0				
10														
11		契約上の	最低販売数	2500	4000	2000	1200	700				[J4]	= SUMPRODUCT(D4:H4, D$14:H$14)	
12		契約上の	最高販売数	5000	5500	4000	3000	3500					→[J4]をコピーし，[J5:J9]へ貼り付け	
13				x_A	x_B	x_C	x_D	x_E				[L2]	= J9 - J4 - L4	
14			生産量											

4. 生産計画

- 例題：結果

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	生産計画													
2			1単位あたり								max.	¥22,116,889	目的関数	
3			製品	A	B	C	D	E		計				
4			変動費	¥500	¥550	¥570	¥490	¥400		¥5,544,222.2		¥5,000,000	固定費（共通）	
5			材料P	1.1	0.0	0.0	1.2	0.9		4,900	≦	4,900	材料P所持量	
6			材料Q	0.0	0.8	0.9	0.1	0.0		5,200	≦	5,200	材料Q所持量	
7			材料R	0.1	0.6	0.4	0.0	0.7		4,038	≦	4,200	材料R所持量	
8			製造時間(h)	0.75	1.10	1.20	1.50	2.00		12,159	≦	14,000	工場総稼働時間(h)	
9			販売価格	¥2,000	¥3,000	¥3,500	¥4,000	¥4,500		¥32,661,111.1				
10														
11		契約上の	最低販売数	2500	4000	2000	1200	700				[J4]	= SUMPRODUCT(D4:H4, D$14:H$14)	
12		契約上の	最高販売数	5000	5500	4000	3000	3500					→[J4]をコピーし，[J5:J9]へ貼り付け	
13				x_A	x_B	x_C	x_D	x_E				[L2]	= J9 - J4 - L4	
14			生産量	2500.0	4000.0	2088.9	1200.0	788.9						
15														
16			利益を最大にする各製品の生産個数を求めよ											
17			※各製品は液体なので，製造数が整数である必要はない											

【参考】Python-MIPで解く

- 記述1：係数設定

```
c = [500, 550, 570, 490, 400]   # 変動費
d = 5000000                     # 固定費
p = [2000,3000,3500,4000,4500]  # 販売価格
b = [4900,5200,4200,14000]      # 供給(材料所持量/稼働時間)
A = [[1.1, 0.0, 0.0, 1.2, 0.9],
     [0.0, 0.8, 0.9, 0.1, 0.0],
     [0.1, 0.6, 0.4, 0.0, 0.7],
     [0.75,1.10,1.20,1.50,2.00]]
lw = [2500,4000,2000,1200, 700] # 販売下限
up = [5000,5500,4000,3000,3500] # 販売上限
I, J = range(len(b)),range(len(c))
```

- 記述2：定式化と求解

```
from mip.model import *

m = Model("ProductEx1")  # モデルの設定

x = [m.add_var(var_type="C", lb=0) for j in J]
m.objective = maximize(xsum((p[j]-c[j])*x[j] for j in J)-d)
for i in I:
  m += xsum(A[i][j]*x[j] for j in J) <= b[i]
for j in J:
  m += x[j] >= lw[j]
  m += x[j] <= up[j]

m.optimize()   # 最適化（求解）の実行

if m.status.value==0:   # もし，最適解が求まったなら
  print("最適解:")     # 最適解を表示
  for j in J:
    print(" x[",j,"] = ", x[j].x)
  print("最適値:", m.objective_value, "=", m.objective) # 目
else:                   # もし，最適解が求まらなかったなら
  print("error:最適解は求まりませんでした")  # エラーメッセ
```

```
最適解:
 x[ 0 ] =  2500.0
 x[ 1 ] =  4000.0
 x[ 2 ] =  2088.888888888889
 x[ 3 ] =  1200.0000000000002
 x[ 4 ] =  788.8888888888879
最適値: 22116888.888888884 = + 1500.0var(0) + 2450.0var(1) +
```

5. 生産スケジューリング

- 例題

10の生産工程からなる製品を製造したい．各工程にかかる時間と，各工程の先行工程（その工程が終わらないと当該工程を開始できない）は表の通り．また，工程間には，0.10の余裕時間をとる．各工程の最早開始時間とスケジュールを求めたい

生産工程	生産時間	先行工程
1	2.33	-
2	2.21	1
3	2.05	1
4	1.50	1
5	5.11	2,3
6	5.14	2,4
7	4.36	3
8	2.84	5,7
9	4.77	6
10	5.72	6,7,8

【変数設定】 生産工程 i の開始時間を変数 t_i とする　　$i \in \{1,2,\ldots,10\}$

5. 生産スケジューリング

- 例題：定式化

$$\text{min.} \quad t_1 + t_2 + \cdots + t_{10}$$

← 各工程最早開始時間最小化 &スケジュール確定

$$\begin{aligned}
\text{s.t.} \quad & t_2 - t_1 \geq 2.33 + 0.10 \\
& t_3 - t_1 \geq 2.33 + 0.10 \\
& t_4 - t_1 \geq 2.33 + 0.10 \\
& t_5 - t_2 \geq 2.21 + 0.10 \\
& t_5 - t_3 \geq 2.05 + 0.10 \\
& t_6 - t_2 \geq 2.33 + 0.10 \\
& t_6 - t_4 \geq 1.50 + 0.10 \\
& t_7 - t_3 \geq 2.05 + 0.10 \\
& t_8 - t_5 \geq 5.11 + 0.10 \\
& t_8 - t_7 \geq 4.36 + 0.10 \\
& t_9 - t_6 \geq 5.14 + 0.10 \\
& t_{10} - t_6 \geq 5.14 + 0.10 \\
& t_{10} - t_7 \geq 4.36 + 0.10 \\
& t_{10} - t_8 \geq 2.84 + 0.10
\end{aligned}$$

生産工程の先行関係条件

$$t_1, \ldots, t_{10} \geq 0$$

← 開始時間は非負

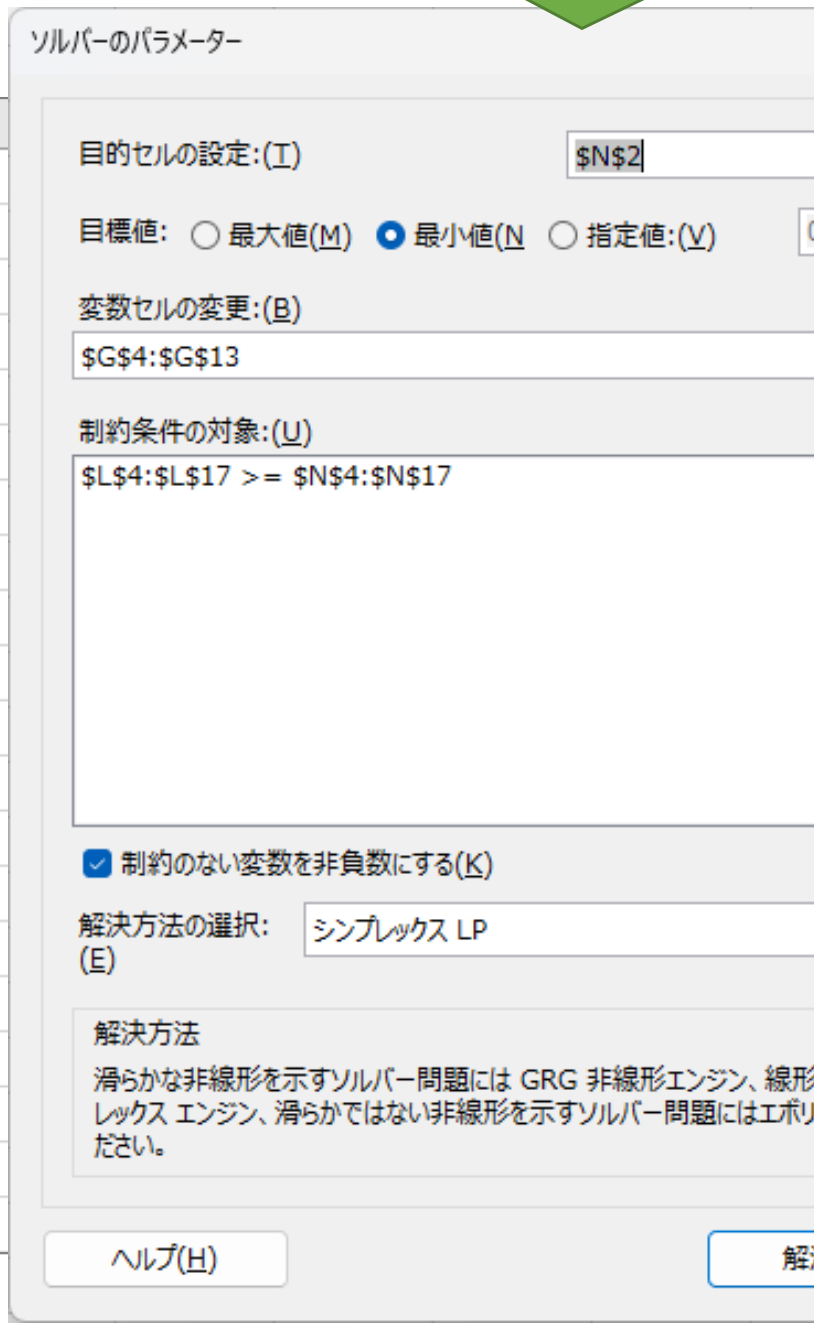
生産工程	生産時間	先行工程
1	2.33	-
2	2.21	1
3	2.05	1
4	1.50	1
5	5.11	2,3
6	5.14	2,4
7	4.36	3
8	2.84	5,7
9	4.77	6
10	5.72	6,7,8

5. 生産スケジューリング

- 例題：Excel記述&ソルバー設定

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	生産スケジューリング													目的関数
2									先行関係				min.	0.00
3		生産工程	生産時間	先行工程			開始時間		先	後		先行関係制約		
4		1	2.33	-		t_1			1	2		0.0	≧	2.43
5		2	2.21	1		t_2			1	3		0.0	≧	2.43
6		3	2.05	1		t_3			1	4		0.0	≧	2.43
7		4	1.50	1		t_4			2	5		0.0	≧	2.31
8		5	5.11	2,3		t_5			3	5		0.0	≧	2.15
9		6	5.14	2,4		t_6			2	6		0.0	≧	2.31
10		7	4.36	3		t_7			4	6		0.0	≧	1.60
11		8	2.84	5,7		t_8			3	7		0.0	≧	2.15
12		9	4.77	6		t_9			5	8		0.0	≧	5.21
13		10	5.72	6,7,8		t_{10}			7	8		0.0	≧	4.46
14									6	9		0.0	≧	5.24
15		余裕時間	0.10						6	10		0.0	≧	5.24
16									7	10		0.0	≧	4.46
17		[N2]	= SUM(G4:G13)						8	10		0.0	≧	2.94
18		[L4]	= INDEX($G$4:$G$13, J4) - INDEX($G$4:$G$13, I4)											
19		[N4]	= VLOOKUP(I4, $B$4:$C$13, 2, FALSE) + $C$15											
20			→[L4:N4]をコピーし，[L5:N17]へ貼り付け											

ソルバーの設定が全て終わったところ



5. 生産スケジューリング

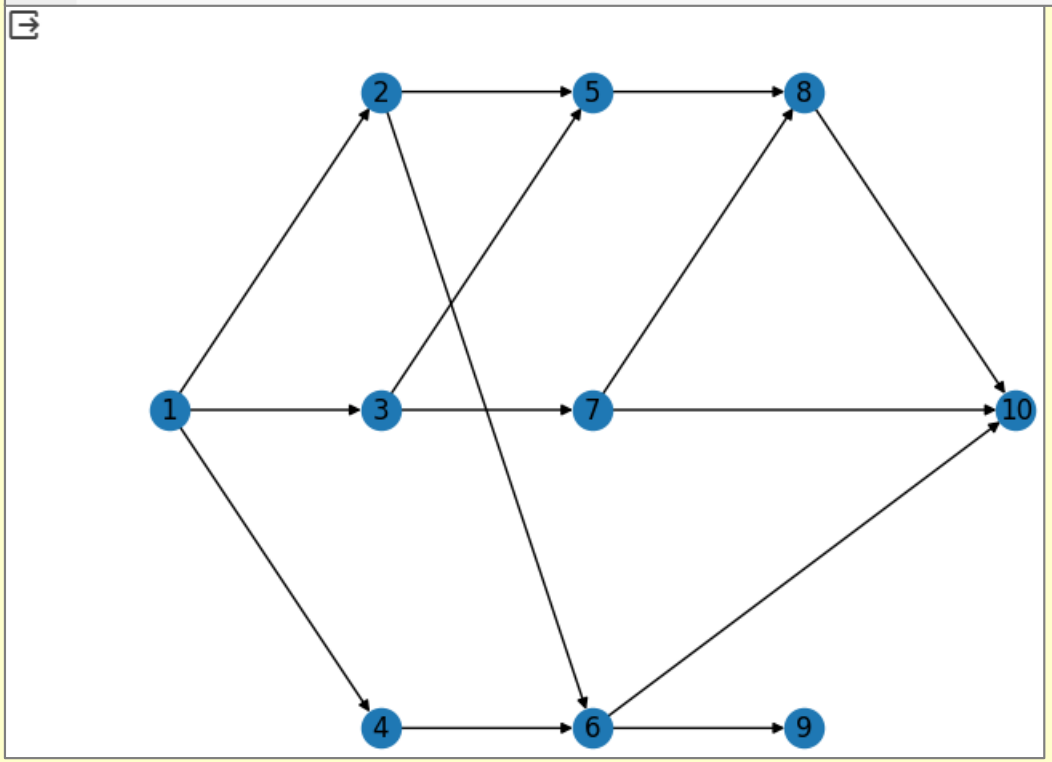
- 例題：結果

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	生産スケジューリング													目的関数
2									先行関係				min.	54.17
3		生産工程	生産時間	先行工程			開始時間		先	後		先行関係制約		
4		1	2.33	-		t_1	0.00		1	2		2.43	≧	2.43
5		2	2.21	1		t_2	2.43		1	3		2.43	≧	2.43
6		3	2.05	1		t_3	2.43		1	4		2.43	≧	2.43
7		4	1.50	1		t_4	2.43		2	5		2.31	≧	2.31
8		5	5.11	2,3		t_5	4.74		3	5		2.31	≧	2.15
9		6	5.14	2,4		t_6	4.74		2	6		2.31	≧	2.31
10		7	4.36	3		t_7	4.58		4	6		2.31	≧	1.60
11		8	2.84	5,7		t_8	9.95		3	7		2.15	≧	2.15
12		9	4.77	6		t_9	9.98		5	8		5.21	≧	5.21
13		10	5.72	6,7,8		t_{10}	12.89		7	8		5.37	≧	4.46
14									6	9		5.24	≧	5.24
15		余裕時間	0.10						6	10		8.15	≧	5.24
16									7	10		8.31	≧	4.46
17		[N2]	= SUM(G4:G13)						8	10		2.94	≧	2.94
18		[L4]	= INDEX($G$4:$G$13, J4) - INDEX($G$4:$G$13, I4)											
19		[N4]	= VLOOKUP(I4, $B$4:$C$13, 2, FALSE) + $C$15											
20			→[L4:N4]をコピーし，[L5:N17]へ貼り付け											

【参考】Python-MIPで解く

- 記述1：係数設定，生産工程の先行関係を表すグラフ

```
%matplotlib inline
import networkx as nx

G = nx.DiGraph()                              # 空の有向グラフ作成：生産工程の先行関係を表すグラフ
G.add_nodes_from([1,2,3,4,5,6,7,8,9,10])  # 点集合：設定・追加
G.add_edges_from([(1,2),(1,3),(1,4),(2,5),(3,5),(2,6),(4,6),(3,7),(5,8),(7,8),(6,9),(6,10),(7,10),(8,10)]) #枝集合：設定・追加
pos = {1:(0,2), 2:(1,3), 3:(1,2), 4:(1,1), 5:(2,3), 6:(2,1), 7:(2,2), 8:(3,3), 9:(3,1), 10:(4,2)} # 点の位置座標設定
nx.draw(G, pos, with_labels=True)
c = [2.33, 2.21, 2.05, 1.50, 5.11, 5.14, 4.36, 2.84, 4.77, 5.72]  # 各点(生産工程)の生産時間
I = range(len(c))
r = 0.10  # 工程間の余裕時間
```



【参考】Python-MIPで解く

- 記述2：定式化と求解

定式化

最適解
と
最適値
の表示

```
from mip.model import *

m = Model("SchedEx1")  # モデルの設定：生産スケジューリング

t = [m.add_var(var_type="C", lb=0) for i in I] # 変数宣言：モデル m に変数を追加
m.objective = minimize(xsum(t[i] for i in I))  # 目的関数の設定：モデル m に目的関数を追加
for (i,j) in G.edges():
  m += t[j-1] - t[i-1] >= c[i-1] + r           # 制約条件の設定：モデル m に制約条件を追加

m.optimize()   # 最適化（求解）の実行

if m.status.value==0:   # もし，最適解が求まったなら
  print("最適解:")     # 最適解を表示
  for i in I:
    print(" t[",i,"] = ", t[i].x)
  print("最適値:", m.objective_value, "=", m.objective) # 目的関数値を表示
else:                   # もし，最適解が求まらなかったなら
  print("error:最適解は求まりませんでした")  # エラーメッセージを表示
```

```
最適解:
 t[ 0 ] =  0.0
 t[ 1 ] =  2.43
 t[ 2 ] =  2.43
 t[ 3 ] =  2.43
 t[ 4 ] =  4.74
 t[ 5 ] =  4.74
 t[ 6 ] =  4.58
 t[ 7 ] =  9.95
 t[ 8 ] =  9.98
 t[ 9 ] =  12.889999999999999
最適値: 54.17 = + var(0) + var(1) + var(2) + var(3) + var(4) + var(5) + var(6) + var(7) + var(8) + var(9)
```

6. 零和ゲームの均衡解とLP

Aは利得最大化プレイヤー
Bは損失最小化プレイヤー

- 2人非協力零和ゲーム
  - プレイヤーAの利得行列(プレイヤーBの損失行列)

A \B	s_{B1}	s_{B2}	…	s_{Bn}
s_{A1}	a_{11}	a_{12}	…	a_{1n}
s_{A2}	a_{21}	a_{22}	…	a_{2n}
…	⋮	⋮		⋮
s_{Am}	a_{m1}	a_{m2}	…	a_{mn}

  - プレイヤーAの混合戦略 $\boldsymbol{p} = (p_1, p_2, \ldots, p_m)$
    - $p_1+p_2+\ldots+p_m=1, \quad p_1, p_2, \ldots, p_m \geqq 0$
  - プレイヤーBの混合戦略 $\boldsymbol{q} = (q_1, q_2, \ldots, q_n)$
    - $q_1+q_2+\ldots+q_m=1, \quad q_1, q_2, \ldots, q_n \geqq 0$

6. 零和ゲームの均衡解とLP

- 2人非協力零和ゲームとLP
  - プレイヤーAの利得行列（Bの損失行列）と混合戦略 $\boldsymbol{p}$

$$\begin{array}{c} \\ p_1 \\ p_2 \\ \vdots \\ p_m \end{array}\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

$$\begin{cases} E(\boldsymbol{p}, s_{B_1}) = a_{11}p_1 + a_{21}p_2 + \cdots + a_{m1}p_m \\ E(\boldsymbol{p}, s_{B_2}) = a_{12}p_1 + a_{22}p_2 + \cdots + a_{m2}p_m \\ \vdots \\ E(\boldsymbol{p}, s_{B_n}) = a_{1n}p_1 + a_{2n}p_2 + \cdots + a_{mn}p_m \end{cases}$$

まとめると…

$$\begin{aligned} & \max . u \\ s.t.\quad & a_{11}p_1 + \cdots + a_{m1}p_m \geq u \\ & a_{12}p_1 + \cdots + a_{m2}p_m \geq u \\ & \cdots \\ & a_{1n}p_1 + \cdots + a_{mn}p_m \geq u \\ & p_1 + \cdots + p_m = 1 \\ & p_1, \ \cdots, \ p_m \geq 0 \end{aligned}$$

$$\max_{\boldsymbol{p}} \min\{E(\boldsymbol{p}, s_{B_1}), E(\boldsymbol{p}, s_{B_2}), \cdots, E(\boldsymbol{p}, s_{B_n})\}$$

6. 零和ゲームの均衡解とLP

- 2人非協力零和ゲームとLP
  - プレイヤーBの損失行列（Aの利得行列）と混合戦略 $\boldsymbol{q}$

$$\begin{array}{cccc} q_1 & q_m & \cdots & q_n \\ \left[\begin{array}{cccc} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{array}\right] \end{array}$$

$$\begin{cases} E(s_{A_1}, \boldsymbol{q}) = a_{11}q_1 + a_{12}q_2 + \cdots + a_{1n}q_n \\ E(s_{A_2}, \boldsymbol{q}) = a_{21}q_1 + a_{22}q_2 + \cdots + a_{2n}q_n \\ \vdots \\ E(s_{A_m}, \boldsymbol{q}) = a_{m1}q_1 + a_{m2}q_2 + \cdots + a_{mn}q_n \end{cases}$$

$$\min_{\boldsymbol{q}} \max\left\{E(s_{A_1}, \boldsymbol{q}), E(s_{A_2}, \boldsymbol{q}), \cdots, E(s_{A_m}, \boldsymbol{q})\right\}$$

まとめると…

$$\begin{array}{l} \min . w \\ s.t. \quad a_{11}q_1 + \cdots + a_{1n}q_n \le w \\ \qquad\quad a_{21}q_1 + \cdots + a_{2n}q_n \le w \\ \qquad\quad \cdots \\ \qquad\quad a_{m1}q_1 + \cdots + a_{mn}q_n \le w \\ \qquad\quad q_1 + \cdots + q_n = 1 \\ \qquad\quad q_1, \cdots, q_n \ge 0 \end{array}$$

6. 零和ゲームの均衡解とLP

Aは利得最大化プレイヤー
Bは損失最小化プレイヤー

- 2人非協力零和ゲームとLP

プレイヤーAの最適化問題
（LPの主問題：**P**）

主・双対

プレイヤーBの最適化問題
（LPの双対問題：**D**）

$$\max . u$$
$$s.t. \quad a_{11}p_1 + \cdots + a_{m1}p_m \geq u$$
$$a_{12}p_1 + \cdots + a_{m2}p_m \geq u$$
$$\cdots$$
$$a_{1n}p_1 + \cdots + a_{mn}p_m \geq u$$
$$p_1 + \cdots + p_m = 1$$
$$p_1, \ \cdots, \ p_m \geq 0$$

$$\min . w$$
$$s.t. \quad a_{11}q_1 + \cdots + a_{1n}q_n \leq w$$
$$a_{21}q_1 + \cdots + a_{2n}q_n \leq w$$
$$\cdots$$
$$a_{m1}q_1 + \cdots + a_{mn}q_n \leq w$$
$$q_1 + \cdots + q_n = 1$$
$$q_1, \ \cdots, \ q_n \geq 0$$

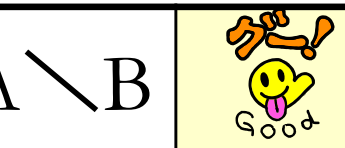
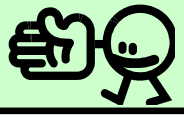
注）（**P**）（**D**）ともに自明解（$\boldsymbol{p}=(1,0,\ldots,0)$, $\boldsymbol{q}=(1,0,\ldots,0)$）があるので実行可能.
→双対定理より，最適解が存在し，最適値は一致する

Theorem

（P），（D）の最適解が（$\boldsymbol{p}^*, u^*$），（$\boldsymbol{q}^*, w^*$）のとき，（$\boldsymbol{p}^*, \boldsymbol{q}^*$）がゲームの均衡点であり，$v := u^* = w^*$がゲームの値である

6. 零和ゲームの均衡解とLP

- 例題：じゃんけん

A＼B				min	max
	0	2	-7	-7	-2
	-2	0	4	-2	
	7	-4	0	-4	
max	7	2	4		
min	2				

マキシミン戦略

ミニマックス戦略

$$-2 = v_1 = \max_i \min_j a_{ij}$$

※

$$2 = v_2 = \min_j \max_i a_{ij}$$

- 両プレイヤーとも，支配戦略は存在しない．
- 純粋戦略ではミニマックス均衡点は存在しない．

6. 零和ゲームの均衡解とLP

- 例題：じゃんけん

		q_1	q_2	q_3
	A＼B			
p_1		0	2	-7
p_2		-2	0	4
p_3		7	-4	0

$$
\begin{array}{l}
\max . u \\
s.t. \quad -2p_2 + 7p_3 \geq u \\
\quad 2p_1 \quad -4p_3 \geq u \\
\quad -7p_1 + 4p_2 \quad \geq u \\
\quad p_1 + p_2 + p_3 = 1 \\
\quad p_1, \quad p_2, \quad p_3 \geq 0
\end{array}
$$

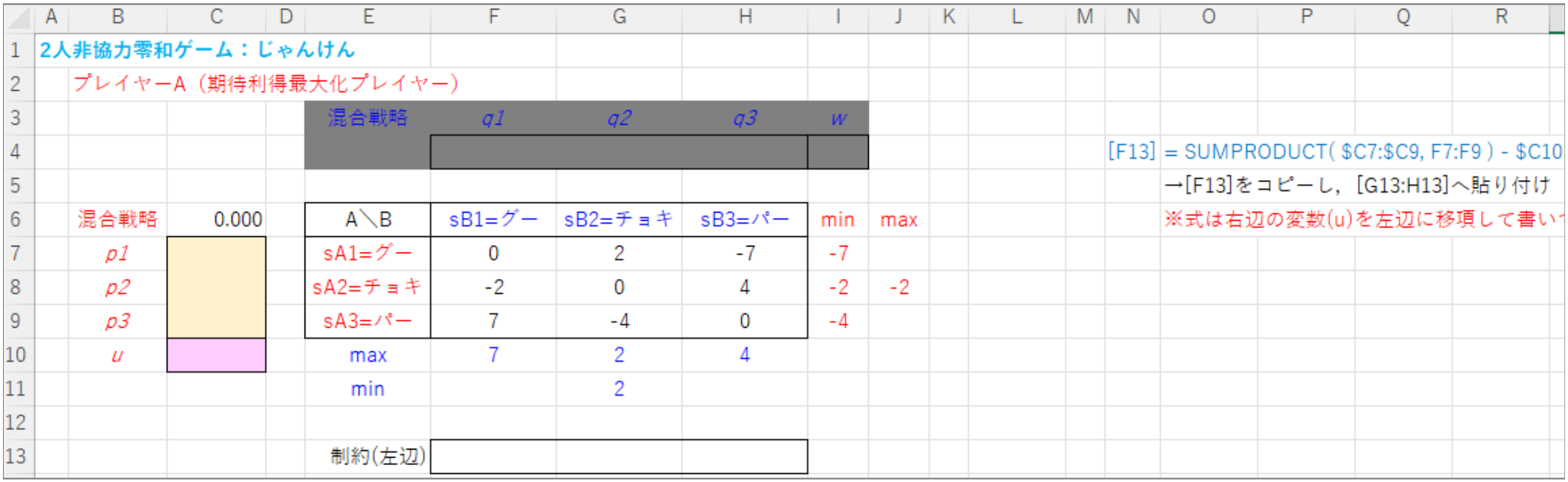
$$
\begin{array}{l}
\min . w \\
s.t. \quad 2q_2 - 7q_3 \leq w \\
\quad -2q_1 \quad +4q_3 \leq w \\
\quad 7q_1 - 4q_2 \quad \leq w \\
\quad q_1 + q_2 + q_3 = 1 \\
\quad q_1, \quad q_2, \quad q_3 \geq 0
\end{array}
$$

自己双対線形計画問題
self-dual LP

【演習】 Excel Solver で最適解を求めよ

6. 零和ゲームの均衡解とLP

- 例題：Excel記述（player A, B を別シートに記述）



2人非協力零和ゲーム：じゃんけん

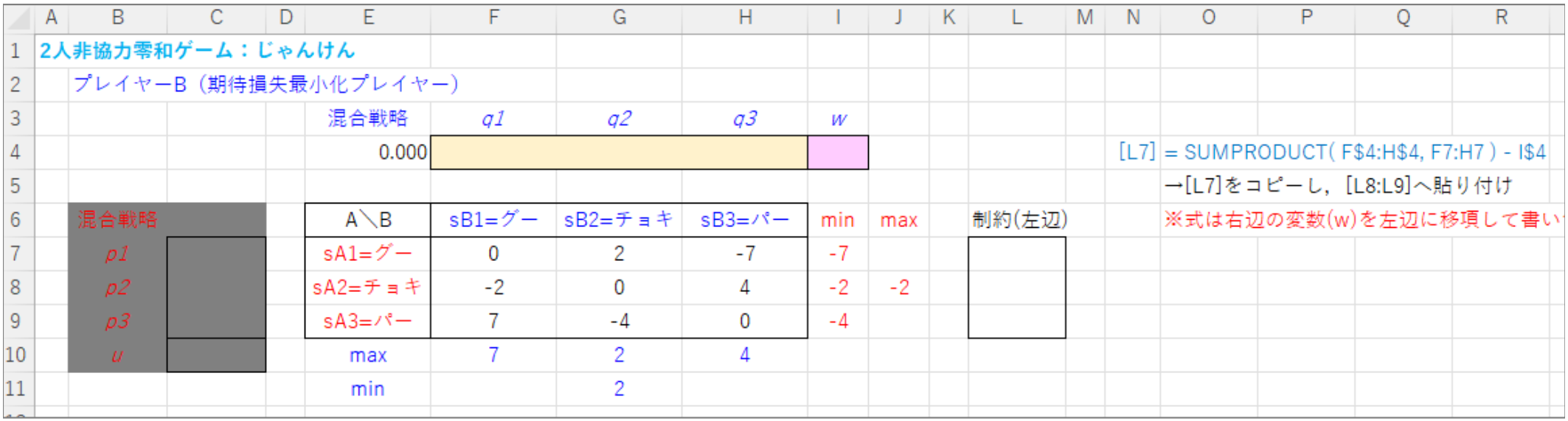
プレイヤーA（期待利得最大化プレイヤー）

	混合戦略	q1	q2	q3	w

[F13] = SUMPRODUCT($C7:$C9, F7:F9) - $C10
→[F13]をコピーし，[G13:H13]へ貼り付け
※式は右辺の変数(u)を左辺に移項して書い

混合戦略	0.000	A＼B	sB1=グー	sB2=チョキ	sB3=パー	min	max
p1		sA1=グー	0	2	-7	-7	
p2		sA2=チョキ	-2	0	4	-2	-2
p3		sA3=パー	7	-4	0	-4	
u		max	7	2	4		
		min		2			

制約(左辺)	



2人非協力零和ゲーム：じゃんけん

プレイヤーB（期待損失最小化プレイヤー）

	混合戦略	q1	q2	q3	w
	0.000				

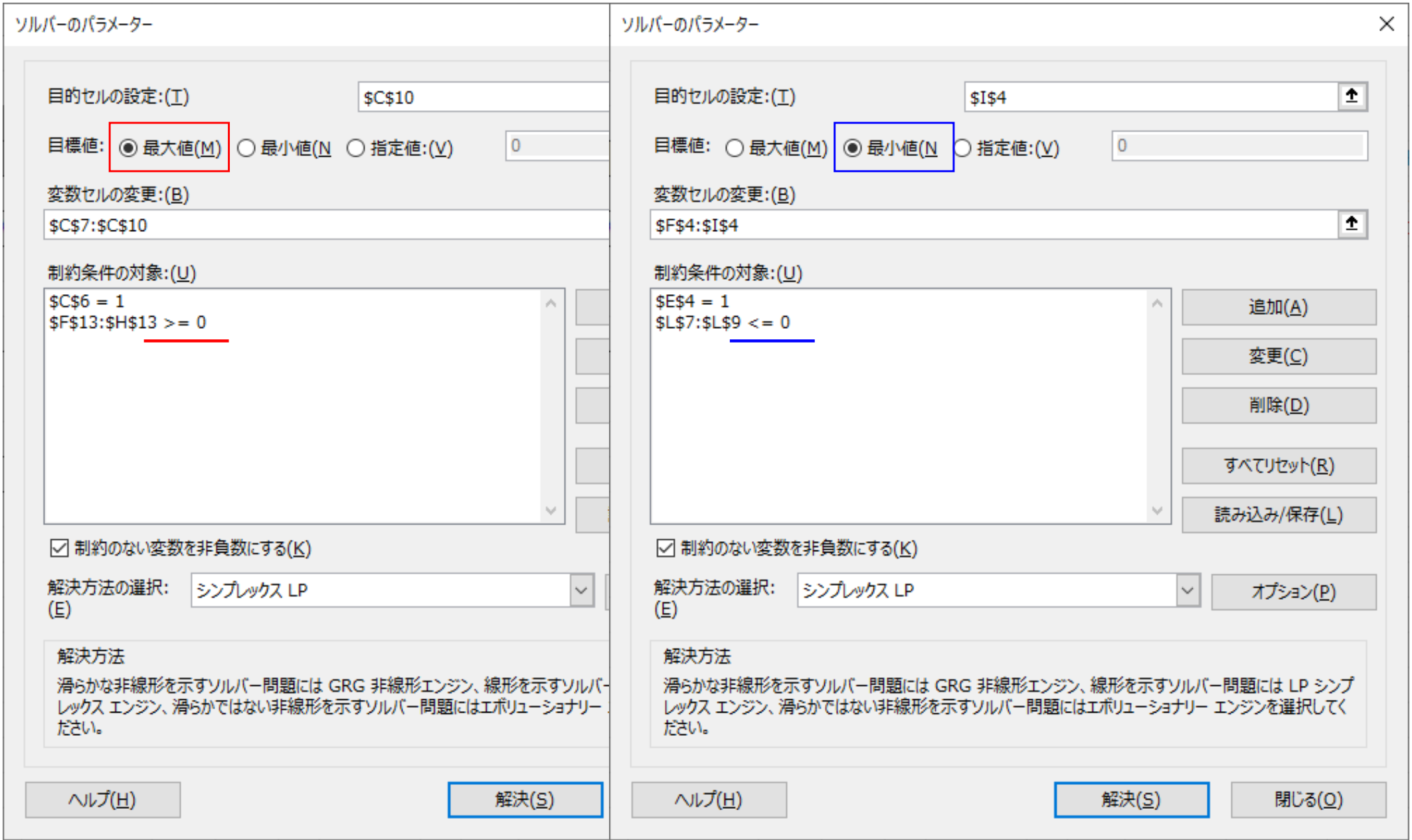
[L7] = SUMPRODUCT(F$4:H$4, F7:H7) - I$4
→[L7]をコピーし，[L8:L9]へ貼り付け
※式は右辺の変数(w)を左辺に移項して書い

混合戦略		A＼B	sB1=グー	sB2=チョキ	sB3=パー	min	max
p1		sA1=グー	0	2	-7	-7	
p2		sA2=チョキ	-2	0	4	-2	-2
p3		sA3=パー	7	-4	0	-4	
u		max	7	2	4		
		min		2			

制約(左辺)

6. 零和ゲームの均衡解とLP

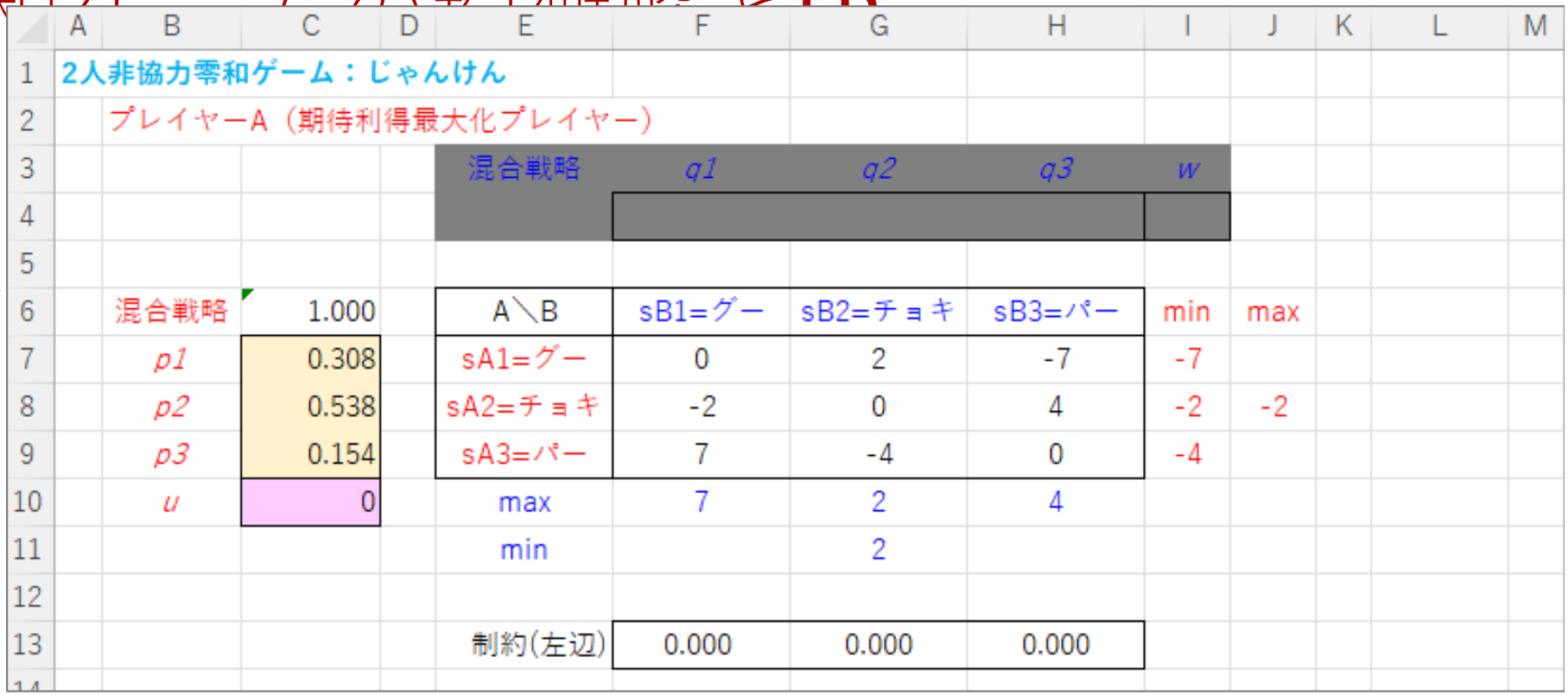
- 例題：ソルバー設定（player A, B を別シートに記述）



6. 零和ゲームの均衡解とLP

- 例題

結果

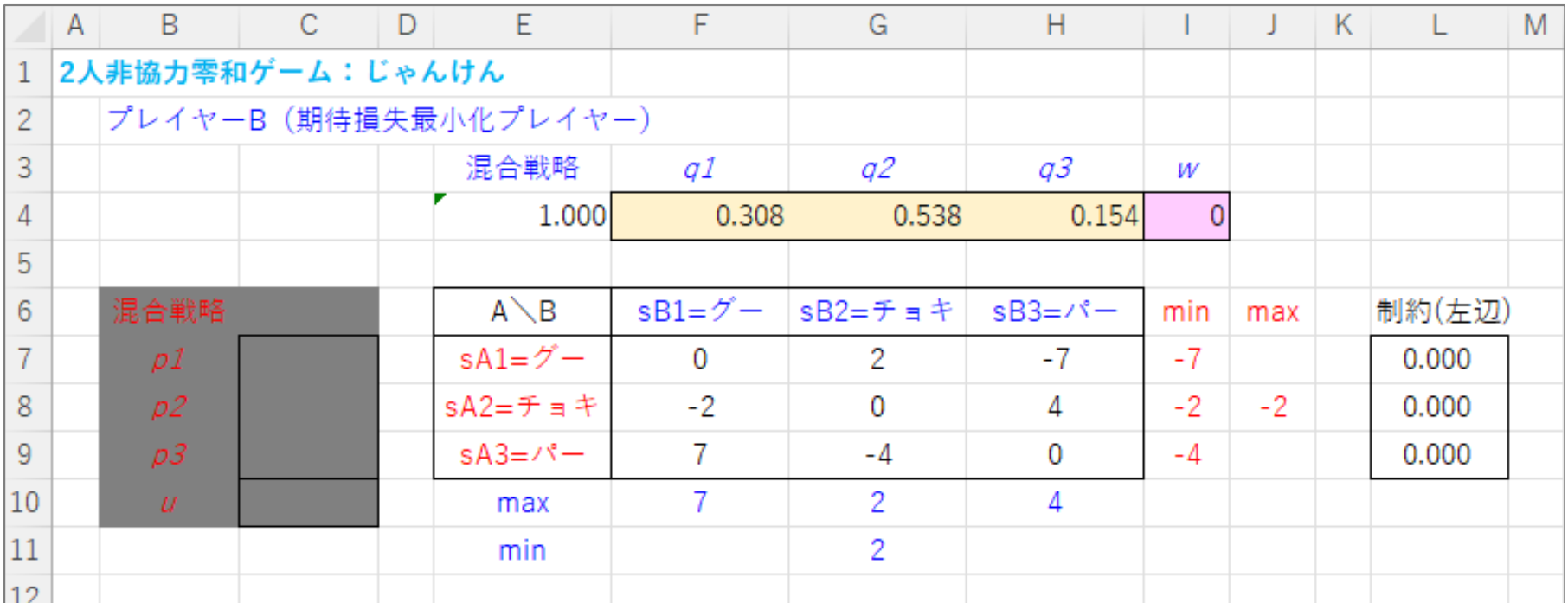


2人非協力零和ゲーム：じゃんけん

プレイヤーA（期待利得最大化プレイヤー）

		混合戦略	q1	q2	q3	w

混合戦略	1.000	A＼B	sB1=グー	sB2=チョキ	sB3=パー	min	max
p1	0.308	sA1=グー	0	2	-7	-7	
p2	0.538	sA2=チョキ	-2	0	4	-2	-2
p3	0.154	sA3=パー	7	-4	0	-4	
u	0	max	7	2	4		
		min		2			
		制約(左辺)	0.000	0.000	0.000		



2人非協力零和ゲーム：じゃんけん

プレイヤーB（期待損失最小化プレイヤー）

	混合戦略	q1	q2	q3	w
1.000		0.308	0.538	0.154	0

混合戦略		A＼B	sB1=グー	sB2=チョキ	sB3=パー	min	max	制約(左辺)
p1		sA1=グー	0	2	-7	-7		0.000
p2		sA2=チョキ	-2	0	4	-2	-2	0.000
p3		sA3=パー	7	-4	0	-4		0.000
u		max	7	2	4			
		min		2				

7. 割当問題

- 例題

✓ 上司は10人の部下（4人は新人で6人はベテラン）に仕事をまかせたい
✓ 仕事は全部で15種類ある（A,B,C,…,O）
✓ 上司は事前に，各仕事に対する部下の成果を5段階評価している（下表）
✓ ベテランは同時に2つまで仕事をまかせられる
✓ 新人は同時に1つまでしか仕事をまかせられず，どの仕事も最大評価は3
✓ 評価値の総和が最大になるように，各部下に仕事を割り振りたい

	社員・仕事	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
新人	1	3	1	1	2	2	2	1	3	2	2	2	1	1	3	3
新人	2	1	2	2	2	1	3	1	3	2	1	2	3	3	2	1
新人	3	1	1	3	2	2	3	3	1	3	3	3	3	1	3	1
新人	4	1	3	2	2	1	1	3	3	2	2	3	3	1	2	3
ベテラン	5	3	2	5	5	2	5	4	4	2	5	2	1	2	5	2
ベテラン	6	4	4	3	4	4	2	4	4	4	4	5	1	4	5	2
ベテラン	7	4	5	3	2	5	2	5	5	5	2	5	2	1	3	3
ベテラン	8	1	2	4	5	4	4	1	5	1	5	5	5	5	1	5
ベテラン	9	4	2	3	5	2	2	4	3	2	2	2	4	3	5	4
ベテラン	10	4	5	5	1	3	2	4	1	1	5	2	5	1	3	5

【変数設定】 0-1変数 x_{ij}：部下 i に仕事 j を割当時1，非割当時0

7. 割当問題

- 例題：定式化

$$\max. \quad \sum_{i=1}^{10}\sum_{j=1}^{15} e_{ij}x_{ij}$$

※ e_{ij}：上司による部下 i の仕事 j の評価予測

$$\text{s.t.} \quad x_{1A} + x_{1B} + x_{1C} + \cdots + x_{1O} \leq 1$$
$$\cdots$$
$$x_{10A} + x_{10B} + x_{10C} + \cdots + x_{10O} \leq 2 \quad \text{①}$$
$$x_{1A} + x_{2A} + \cdots + x_{10A} = 1$$
$$\cdots$$
$$x_{1O} + x_{2O} + \cdots + x_{10O} = 1 \quad \text{②}$$
$$x_{1A}, \ldots, x_{10O} \in \{0,1\}$$

制約①：部下(新人 or ベテラン)の担当仕事数は 1 or 2 以下
制約②：各仕事は必ず誰かが担当する
0-1制約は $0 \leqq x_{ij} \leqq 1$ にして解いてよい（補足：完全単模行列）

【補足】

- 単模行列(unimodular matrix)
  - def) 整数正方行列$A \in R^{n \times n}$が単模行列　⇔　$\det A = 1$ or -1
  - theorem) 単模行列の逆行列も，整数行列で単模行列

- 完全単模行列(totally unimodular matrix)
  - def) 整数行列$A \in R^{m \times n}$が完全単模行列
    ⇔　任意の小行列式の値が 0 or 1 or -1
  - theorem) 完全単模行列の各要素は 0 or 1 or -1
  - ex) 有向グラフの接続行列は完全単模
  - ex) 無向グラフの接続行列が完全単模となる必要十分条件はグラフが2部であること（cf. 3点奇数サイクルのグラフは$\det A = \pm 2$）
  - theorem) LP(P)が最適解をもち，係数行列Aが完全単模とする．bが整数ベクトルなら，(P)は整数最適解 $x \in Z^n$をもつ

$$(P) \quad max.\ c^t x \quad s.t. \quad Ax = b,\ x \geqq 0$$

proof) (P)の基底Bに対する基底解は$(B^{-1}b, 0)$
Aが完全単模なので，Bは単模行列
よって，B^{-1}は整数行列．故に$B^{-1}b$は整数ベクトル■

7. 割当問題 • 例題：Excel記述

割当問題　上司が予想する出来具合（5段階評価）

社員・仕事	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
新人 1	3	1	1	2	2	2	1	3	2	2	2	1	1	3	3
新人 2	1	2	2	2	1	3	1	3	2	1	2	3	3	2	1
新人 3	1	1	3	2	2	3	3	1	3	3	3	3	1	3	1
新人 4	1	3	2	2	1	1	3	3	2	2	3	3	1	2	3
ベテラン 5	3	2	5	5	2	5	4	4	2	5	2	1	2	5	2
ベテラン 6	4	4	3	4	4	2	4	4	4	4	5	1	4	5	2
ベテラン 7	4	5	3	2	5	2	5	5	5	2	5	2	1	3	3
ベテラン 8	1	2	4	5	4	4	1	5	1	5	5	5	5	1	5
ベテラン 9	4	2	3	5	2	2	4	3	2	2	2	4	3	5	4
ベテラン 10	4	5	5	1	3	2	4	1	1	5	2	5	1	3	5

[T15] = SUM(D15:R15)
→[T15]をコピーし，[T16:T24]へ貼り付け
[D26] = SUM(D15:D24)
→[D26]をコピーし，[E26:R26]へ貼り付け
[D30] = SUMPRODUCT(D3:D12, D15:D24)
→[D30]をコピーし，[E30:R30]へ貼り付け
[T30] = SUM(D30:R30)

誰がどの仕事をするかの {0,1}-変数（変化させるセル）

社員・仕事	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	担当数		担当可能数
1																	≦	1
2																	≦	1
3																	≦	1
4																	≦	1
5																	≦	2
6																	≦	2
7																	≦	2
8																	≦	2
9																	≦	2
10																	≦	2

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
担当数															
	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=
担当可能数	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

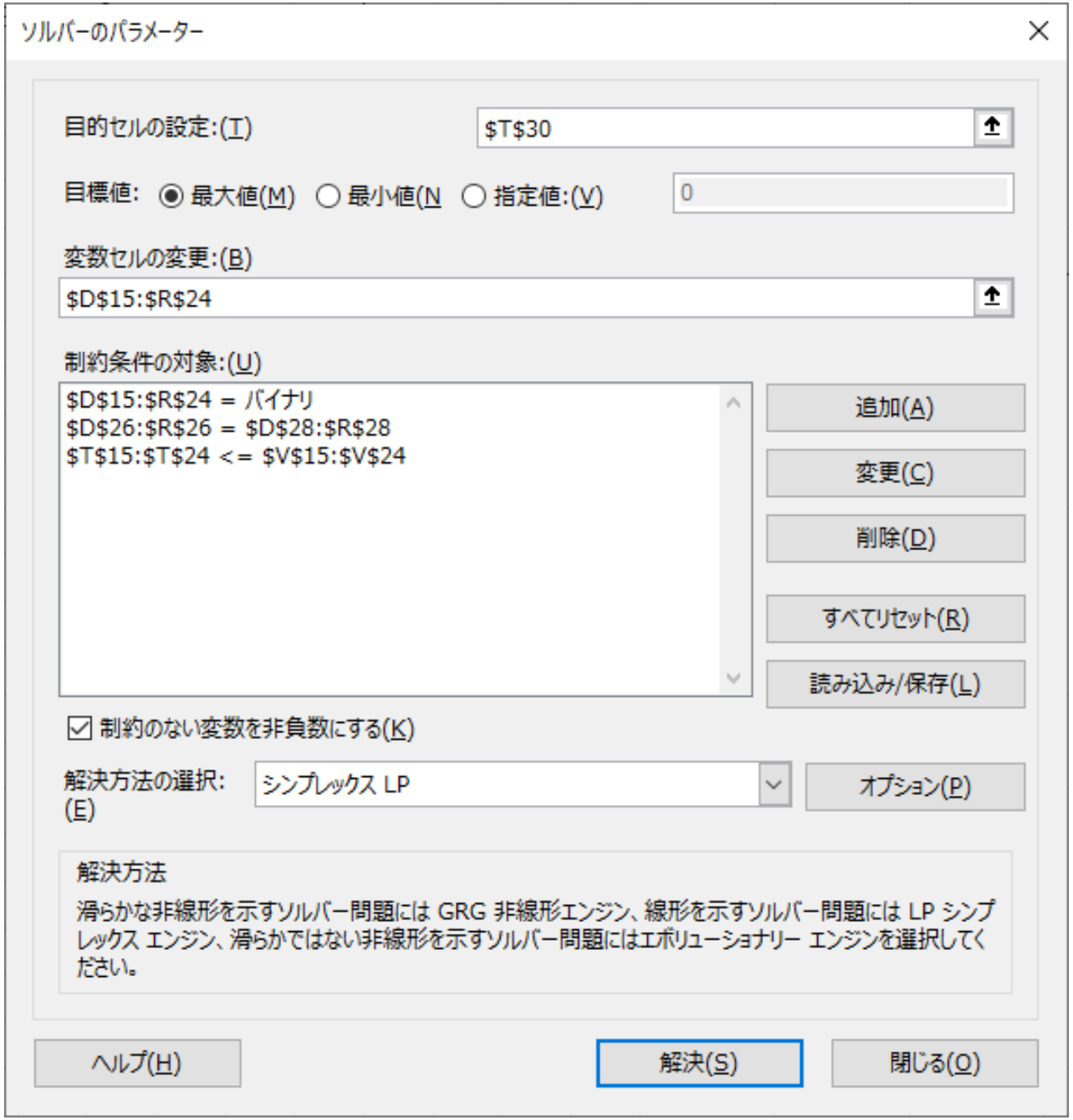
目的関数：最大化

期待出来具合　　総出来具合

7. 割当問題

- 例題：

ソルバー設定



7. 割当問題

- 例題：結果

誰がどの仕事をするかの {0,1}-変数（変化させるセル）

社員・仕事	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	担当数		担当可能数
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	≦	1
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	≦	1
3	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	≦	1
4	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	≦	1
5	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	2	≦	2
6	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	2	≦	2
7	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	2	≦	2
8	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	2	≦	2
9	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	2	≦	2
10	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	2	≦	2
担当数	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1			
	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=			
担当可能数	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1			
期待出来具合	3	5	3	5	5	5	3	4	5	5	5	5	5	5	5	目的関数：最大化		
																68		総出来具合

7. 割当問題

- 例題：結果2（全員が1つ以上の仕事を担当する場合）

誰がどの仕事をするかの {0,1}-変数（変化させるセル）

社員・仕事	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	担当数		担当可能数
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	≦	1
2	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	≦	1
3	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	≦	1
4	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	≦	1
5	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	2	≦	2
6	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	2	≦	2
7	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	2	≦	2
8	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	2	≦	2
9	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	≦	2
10	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	2	≦	2
担当数	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1			
	=	=	=	=	=	=	=	=	=	=	=	=	=	=	=			
担当可能数	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1			
																目的関数：最大化		
持出来具合	3	3	5	5	5	5	5	5	3	5	5	5	3	5	5	67		総出来具合

ソルバーのパラメーター

目的セルの設定:(T) $T$30

目標値: ◉ 最大値(M) ○ 最小値(N ○ 指定値

変数セルの変更:(B)

$D$15:$R$24

制約条件の対象:(U)

$D$15:$R$24 = バイナリ
$D$26:$R$26 = $D$28:$R$28
$T$15:$T$24 <= $V$15:$V$24
$T$15:$T$24 >= 1

☑ 制約のない変数を非負数にする(K)

解決方法の選択:(E) シンプレックス LP

解決方法

【参考】Python-MIPで解く

- 記述1：係数設定

```
E = [[3,1,1,2,2,2,1,3,2,2,2,1,1,3,3],
     [1,2,2,2,1,3,1,3,2,1,2,3,3,2,1],
     [1,1,3,2,2,3,3,1,3,3,3,3,1,3,1],
     [1,3,2,2,1,1,3,3,2,2,3,3,1,2,3],
     [3,2,5,5,2,5,4,4,2,5,2,1,2,5,2],
     [4,4,3,4,4,2,4,4,4,4,5,1,4,5,2],
     [4,5,3,2,5,2,5,5,5,2,5,2,1,3,3],
     [1,2,4,5,4,4,1,5,1,5,5,5,5,1,5],
     [4,2,3,5,2,2,4,3,2,2,2,4,3,5,4],
     [4,5,5,1,3,2,4,1,1,5,2,5,1,3,5]]
b = [1,1,1,1,2,2,2,2,2,2]
I,J = range(len(b)),range(15)
```

- 記述2：定式化と求解

定式化

```
x = m.add_var_tensor((10,15), name="x", var_type="B") # 0-1変数(size=|I|x|J|)
m.objective = maximize(xsum(E[i][j]*x[i][j] for i in I for j in J)) # 目的関数
for i in I:
  m += xsum(x[i][j] for j in J) <= b[i] # 制約1：担当数は担当可能数以下
for j in J:
  m += xsum(x[i][j] for i in I) == 1    # 制約2：各仕事は誰かが必ず担当

m.optimize()   # 最適化（求解）の実行
```

最適解と最適値の表示

```
if m.status.value==0: # もし，最適解が求まったなら
  print("最適解:")    # 最適解を表示
  for i in I:
    for j in J:
      if x[i][j].x == 1:
        print(" x[",i+1,j+1,"]=", x[i][j].x, end=" ")
    print()
  print("最適値:", m.objective_value, "=", m.objective) # 目的関数値を表示
else:                 # もし，最適解が求まらなかったなら
  print("error:最適解は求まりませんでした")            # エラーメッセージ表示
```

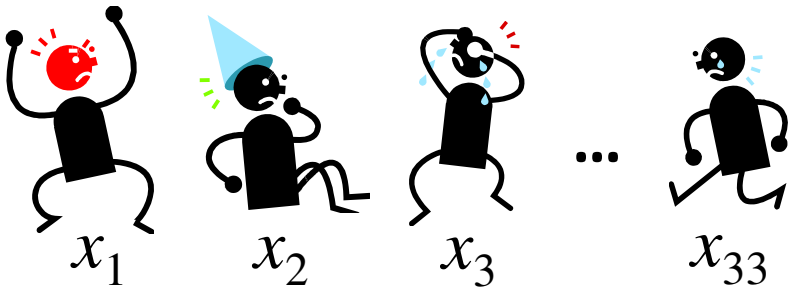
```
最適解:
 x[ 1 1 ]= 1.0
 x[ 2 12 ]= 1.0
 x[ 3 7 ]= 1.0

 x[ 5 6 ]= 1.0  x[ 5 10 ]= 1.0
 x[ 6 9 ]= 1.0  x[ 6 11 ]= 1.0
 x[ 7 2 ]= 1.0  x[ 7 5 ]= 1.0
 x[ 8 8 ]= 1.0  x[ 8 13 ]= 1.0
 x[ 9 4 ]= 1.0  x[ 9 14 ]= 1.0
 x[ 10 3 ]= 1.0  x[ 10 15 ]= 1.0
最適値: 68.0 = + 3.0x_0_0 + x_0_1 + x_0_2 + 2.0x_0_3 + 2.0x_0_4 + 2.0x_0_5 + x
```

8. クラス編成問題

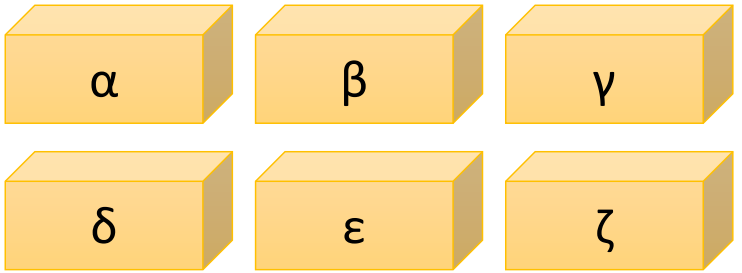
- 例題

✓ 33人の学生を6クラスに配属させたい
✓ 各学生は丁度1つのクラスに所属させ，配属しないという選択はないとする
✓ 各学生は6クラスへの希望を持っている（第1, 2, 3志望と志望外）
　→ 配属時満足度：第1志望100点 / 第2志望60点 / 第3志望30点 / 志望外-999点
✓ クラスには定員があり，全て6人である（容量6人×6クラス=36人で充分）
✓ 全学生の満足度総和が最大になるように学生をクラスへ配属させなさい

学生33人



6クラス（各定員6人）



【変数設定】 0-1変数 x_{ij}：学生 i をクラス j に所属時1，非所属時0

8. クラス編成問題

- 例題：定式化

$$\text{max.} \quad \sum_{j=1}^{6}\sum_{i=1}^{33} u_{ij}x_{ij}$$

※ u_{ij}：学生 i のクラス j 配属時満足度

$$\text{s.t.} \quad \begin{matrix} x_{1\alpha} + x_{1\beta} + x_{1\gamma} + \cdots + x_{1\zeta} = 1 \\ \cdots \\ x_{33\alpha} + x_{33\beta} + x_{33\gamma} + \cdots + x_{33\zeta} = 1 \end{matrix} \quad \left.\right\} \text{①}$$

$$\begin{matrix} x_{1\alpha} + x_{2\alpha} + \cdots + x_{33\alpha} \leq 6 \\ \cdots \\ x_{1\zeta} + x_{2\zeta} + \cdots + x_{33\zeta} \leq 6 \end{matrix} \quad \left.\right\} \text{②}$$

$$x_{1\alpha}, \ldots, x_{33\zeta} \in \{0,1\}$$

制約①：各学生は6クラスのどこか1か所に所属する
制約②：各クラスの定員は6人
0-1制約は $0 \leqq x_{ij} \leqq 1$ にして解いてよい（補足：完全単模行列）

8. クラス編成問題

- 例題： Excel記述

クラス編成問題

学生志望			効用
第	1	志望	100
第	2	志望	60
第	3	志望	30

定員	6 名

学生希望度合計	
学生希望度平均	

各学生のクラス希望順位 / 学生効用（満足度） / 変化させるセル（ソルバー用） / 最適解（配属結果：希望度） / 学生の希望達成度（第○希望か？）

行	B	C	D	E	F	G	氏名	研究室α	研究室β	研究室γ	研究室δ	研究室ε	研究室ζ		α	β	γ	δ	ε	ζ	計		α	β	γ	δ	ε	ζ		
7	4	3	1	5	6	2	学生1							1								1							第	希望
8	2	3	4	6	1	5	学生2							2								2							第	希望
9	6	4	5	3	1	2	学生3							3								3							第	希望
10	5	2	3	4	1	6	学生4							4								4							第	希望
11	4	3	2	6	5	1	学生5							5								5							第	希望
12	1	5	3	6	2	4	学生6							6								6							第	希望
13	1	4	2	3	5	6	学生7							7								7							第	希望
14	1	2	4	3	6	5	学生8							8								8							第	希望
15	4	5	2	3	1	6	学生9							9								9							第	希望
16	5	3	1	4	2	6	学生10							10								10							第	希望
17	5	4	1	3	2	6	学生11							11								11							第	希望
18	4	1	6	2	3	5	学生12							12								12							第	希望
19	5	2	4	6	1	3	学生13							13								13							第	希望
20	1	4	6	2	3	5	学生14							14								14							第	希望
21	5	1	2	4	6	3	学生15							15								15							第	希望
22	1	5	3	2	4	6	学生16							16								16							第	希望
23	5	6	4	2	3	1	学生17							17								17							第	希望
24	6	1	2	5	3	4	学生18							18								18							第	希望
25	2	6	1	5	4	3	学生19							19								19							第	希望
26	6	2	4	1	5	3	学生20							20								20							第	希望
27	1	2	6	3	5	4	学生21							21								21							第	希望
28	6	3	2	1	4	5	学生22							22								22							第	希望
29	4	2	5	1	6	3	学生23							23								23							第	希望
30	1	3	4	5	6	2	学生24							24								24							第	希望
31	2	6	5	1	3	4	学生25							25								25							第	希望
32	2	3	5	6	1	4	学生26							26								26							第	希望
33	2	6	1	3	5	4	学生27							27								27							第	希望
34	6	5	1	2	4	3	学生28							28								28							第	希望
35	4	3	2	6	5	1	学生29							29								29							第	希望
36	4	1	5	3	6	2	学生30							30								30							第	希望
37	3	2	5	6	4	1	学生31							31								31							第	希望
38	4	6	3	5	2	1	学生32							32								32							第	希望
39	3	1	2	5	4	6	学生33							33								33							第	希望
40														計																

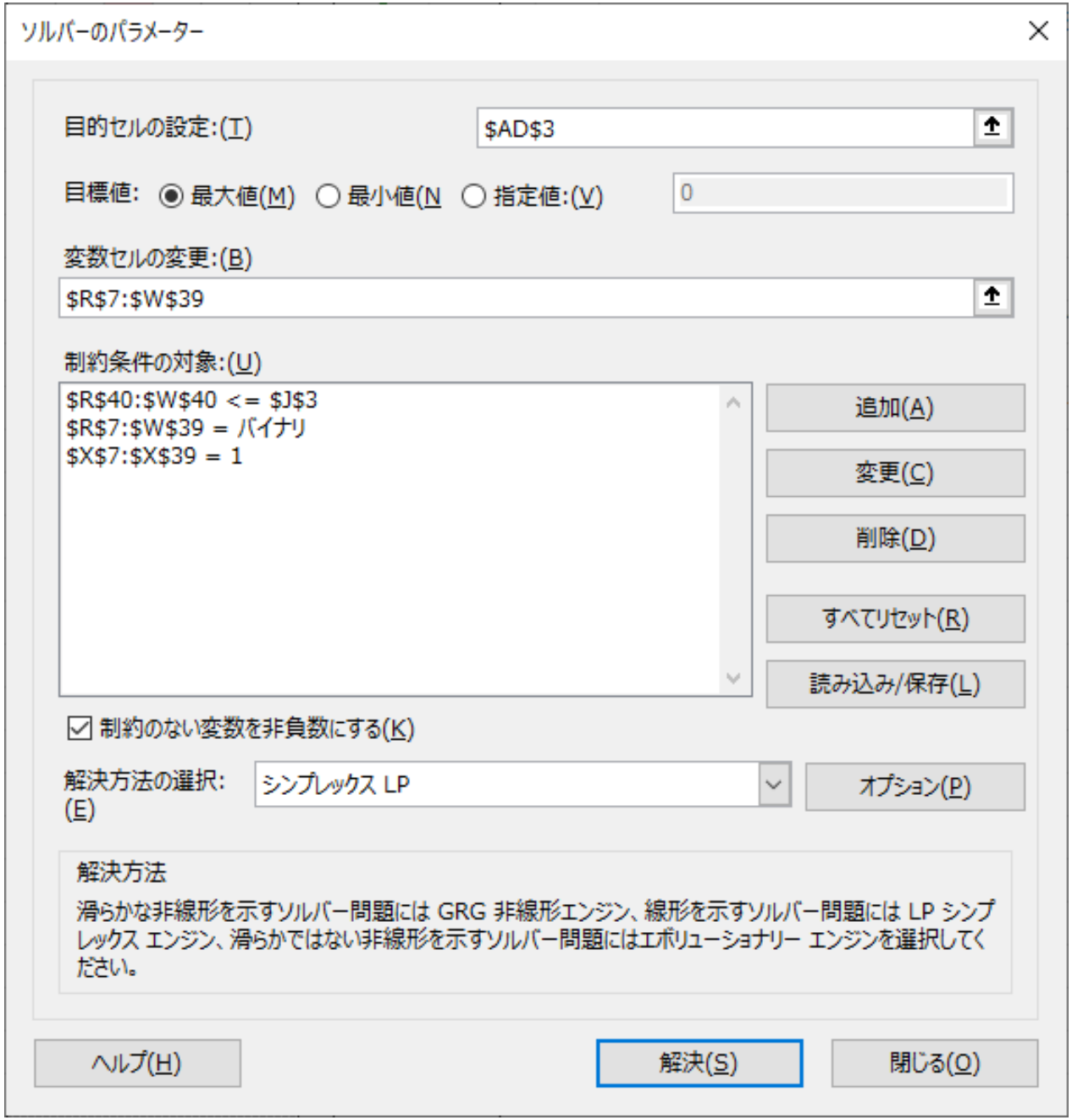
[J7] = IF(ISNA(VLOOKUP(B7, $D$3:$F$5, 3, FALSE)),
-999,
VLOOKUP(B7, $D$3:$F$5, 3, FALSE))
—[J7]をコピーし，[J7:O39]へ貼り付け
[X7] = SUM(R7:W7)
—[X7]をコピーし，[X8:X39]へ貼り付け
[R40] = SUM(R7:R39)
—[R40]をコピーし，[S40:W40]へ貼り付け
[AA7] = J7 * R7
—[AA7]をコピーし，[AA7:AF39]へ貼り付け
[AD3] = SUM(AA7:AF39)
[AD4] = AD3 / 33

[AI7] = SUMPRODUCT(B7:G7, R7:W7)
—[AI7]をコピーし，[AI8:AI39]へ貼り付け

8. クラス編成問題

- 例題：

ソルバー設定



8. ク

- 例題

結果

クラス編成問題

学生志望			効用
第	1	志望	100
第	2	志望	60
第	3	志望	30

定員 6 名

学生希望度合計 3260

学生希望度平均 98.79

学生効用（満足度）

変化させるセル（ソルバー用）

最適解（配属結果：希望度）

学生の希望達成度

各学生のクラス希望順位						氏名	研究室α	研究室β	研究室γ	研究室δ	研究室ε	研究室ζ		α	β	γ	δ	ε	ζ	計		α	β	γ	δ	ε	ζ	（第○希望か？）	
4	3	1	5	6	2	学生1	-999	30	100	-999	-999	60	1	0	0	1	0	0	0	1	1	0	0	100	0	0	0	第	1 希望
2	3	4	6	1	5	学生2	60	30	-999	-999	100	-999	2	0	0	0	0	1	0	1	2	0	0	0	0	100	0	第	1 希望
6	4	5	3	1	2	学生3	-999	-999	-999	30	100	60	3	0	0	0	0	1	0	1	3	0	0	0	0	100	0	第	1 希望
5	2	3	4	1	6	学生4	-999	60	30	-999	100	-999	4	0	0	0	0	1	0	1	4	0	0	0	0	100	0	第	1 希望
4	3	2	6	5	1	学生5	-999	30	60	-999	-999	100	5	0	0	0	0	0	1	1	5	0	0	0	0	0	100	第	1 希望
1	5	3	6	2	4	学生6	100	-999	30	-999	60	-999	6	1	0	0	0	0	0	1	6	100	0	0	0	0	0	第	1 希望
1	4	2	3	5	6	学生7	100	-999	60	30	-999	-999	7	1	0	0	0	0	0	1	7	100	0	0	0	0	0	第	1 希望
1	2	4	3	6	5	学生8	100	60	-999	30	-999	-999	8	1	0	0	0	0	0	1	8	100	0	0	0	0	0	第	1 希望
4	5	2	3	1	6	学生9	-999	-999	60	30	100	-999	9	0	0	0	0	1	0	1	9	0	0	0	0	100	0	第	1 希望
5	3	1	4	2	6	学生10	-999	30	100	-999	60	-999	10	0	0	1	0	0	0	1	10	0	0	100	0	0	0	第	1 希望
5	4	1	3	2	6	学生11	-999	-999	100	30	60	-999	11	0	0	1	0	0	0	1	11	0	0	100	0	0	0	第	1 希望
4	1	6	2	3	5	学生12	-999	100	-999	60	30	-999	12	0	1	0	0	0	0	1	12	0	100	0	0	0	0	第	1 希望
5	2	4	6	1	3	学生13	-999	60	-999	-999	100	30	13	0	0	0	0	1	0	1	13	0	0	0	0	100	0	第	1 希望
1	4	6	2	3	5	学生14	100	-999	-999	60	30	-999	14	0	0	0	1	0	0	1	14	0	0	0	60	0	0	第	2 希望
5	1	2	4	6	3	学生15	-999	100	60	-999	-999	30	15	0	1	0	0	0	0	1	15	0	100	0	0	0	0	第	1 希望
1	5	3	2	4	6	学生16	100	-999	30	60	-999	-999	16	1	0	0	0	0	0	1	16	100	0	0	0	0	0	第	1 希望
5	6	4	2	3	1	学生17	-999	-999	-999	60	30	100	17	0	0	0	0	0	1	1	17	0	0	0	0	0	100	第	1 希望
6	1	2	5	3	4	学生18	-999	100	60	-999	30	-999	18	0	1	0	0	0	0	1	18	0	100	0	0	0	0	第	1 希望
2	6	1	5	4	3	学生19	60	-999	100	-999	-999	30	19	0	0	1	0	0	0	1	19	0	0	100	0	0	0	第	1 希望
6	2	4	1	5	3	学生20	-999	60	-999	100	-999	30	20	0	0	0	1	0	0	1	20	0	0	0	100	0	0	第	1 希望
1	2	6	3	5	4	学生21	100	60	-999	30	-999	-999	21	1	0	0	0	0	0	1	21	100	0	0	0	0	0	第	1 希望
6	3	2	1	4	5	学生22	-999	30	60	100	-999	-999	22	0	0	0	1	0	0	1	22	0	0	0	100	0	0	第	1 希望
4	2	5	1	6	3	学生23	-999	60	-999	100	-999	30	23	0	0	0	1	0	0	1	23	0	0	0	100	0	0	第	1 希望
1	3	4	5	6	2	学生24	100	30	-999	-999	-999	60	24	1	0	0	0	0	0	1	24	100	0	0	0	0	0	第	1 希望
2	6	5	1	3	4	学生25	60	-999	-999	100	30	-999	25	0	0	0	1	0	0	1	25	0	0	0	100	0	0	第	1 希望
2	3	5	6	1	4	学生26	60	30	-999	-999	100	-999	26	0	0	0	0	1	0	1	26	0	0	0	0	100	0	第	1 希望
2	6	1	3	5	4	学生27	60	-999	100	30	-999	-999	27	0	0	1	0	0	0	1	27	0	0	100	0	0	0	第	1 希望
6	5	1	2	4	3	学生28	-999	-999	100	60	-999	30	28	0	0	1	0	0	0	1	28	0	0	100	0	0	0	第	1 希望
4	3	2	6	5	1	学生29	-999	30	60	-999	-999	100	29	0	0	0	0	0	1	1	29	0	0	0	0	0	100	第	1 希望
4	1	5	3	6	2	学生30	-999	100	-999	30	-999	60	30	0	1	0	0	0	0	1	30	0	100	0	0	0	0	第	1 希望
3	2	5	6	4	1	学生31	30	60	-999	-999	-999	100	31	0	0	0	0	0	1	1	31	0	0	0	0	0	100	第	1 希望
4	6	3	5	2	1	学生32	-999	-999	30	-999	60	100	32	0	0	0	0	0	1	1	32	0	0	0	0	0	100	第	1 希望
3	1	2	5	4	6	学生33	30	100	60	-999	-999	-999	33	0	1	0	0	0	0	1	33	0	100	0	0	0	0	第	1 希望
													計	6	5	6	5	6	5										

【参考】Python-MIPで解く

- 記述1:係数設定

```
U = [[-999,30,100,-999,-999,60],  # 各学生のクラス満足度
     [60,30,-999,-999,100,-999],
     [-999,-999,-999,30,100,60],
     [-999,60,30,-999,100,-999],
     [-999,30,60,-999,-999,100],
     [100,-999,30,-999,60,-999],
     [100,-999,60,30,-999,-999],
     [100,60,-999,30,-999,-999],
     [-999,-999,60,30,100,-999],
     [-999,30,100,-999,60,-999],
     [-999,-999,100,30,60,-999],
     [-999,100,-999,60,30,-999],
     [-999,60,-999,-999,100,30],
     [100,-999,-999,60,30,-999],
     [-999,100,60,-999,-999,30],
     [100,-999,30,60,-999,-999],
     [-999,-999,-999,60,30,100],
     [-999,100,60,-999,30,-999],
     [60,-999,100,-999,-999,30],
     [-999,60,-999,100,-999,30],
     [100,60,-999,30,-999,-999],
     [-999,30,60,100,-999,-999],
     [-999,60,-999,100,-999,30],
     [100, 30,-999,-999,-999,60],
     [60,-999,-999,100,30,-999],
     [60,30,-999,-999,100,-999],
     [60,-999,100,30,-999,-999],
     [-999,-999,100,60,-999,30],
     [-999,30,60,-999,-999,100],
     [-999,100,-999,30,-999,60],
     [30,60,-999,-999,-999,100],
     [-999,-999,30,-999,60,100],
     [30,100,60,-999,-999,-999]]
num,cap = 33,6  # 学生数,各クラス定員
I,J = range(num),range(cap)
```

【参考】Python-MIPで解く

- 記述2：定式化と求解

定式化

```
x = m.add_var_tensor((num,cap), name="x", var_type="B") # 0-1変数(size=|I|x|J|)
m.objective = maximize(xsum(U[i][j]*x[i][j] for i in I for j in J)) # 目的関数
for i in I:
  m += xsum(x[i][j] for j in J) == 1   # 制約1：各学生は必ず1クラスに所属
for j in J:
  m += xsum(x[i][j] for i in I) <= cap # 制約2：各クラス所属学生数は定員以下

m.optimize()   # 最適化（求解）の実行
```

最適解と最適値の表示

```
if m.status.value==0: # もし，最適解が求まったなら
  print("最適解:")    # 最適解を表示
  for j in J:
    sum = 0
    print("class", j+1, end=": ")
    for i in I:
      if x[i][j].x == 1:
        sum += 1
        print(i+1, end=" ")
    print("[",sum,"人]")
  print("最適値:", m.objective_value, "=", m.objective) # 目的関数値を表示
else:                  # もし，最適解が求まらなかったなら
  print("error:最適解は求まりませんでした")            # エラーメッセージ表示
```

```
最適解:
class 1: 6 7 14 16 21 24 [ 6 人]
class 2: 8 12 15 18 30 33 [ 6 人]
class 3: 1 10 11 19 27 28 [ 6 人]
class 4: 20 22 23 25 [ 4 人]
class 5: 2 3 4 9 13 26 [ 6 人]
class 6: 5 17 29 31 32 [ 5 人]
最適値: 3260.0 = - 999.0x_0_0 + 30.0x_0_1 + 100.0x_0_2 - 999.0x_0_3 - 999.0x_0_
```

9. 適当な問題を生成して解かせてみる

- 例題：乱数によりLPの問題を生成
  - Excelによる乱数の生成方法
    1. 関数を使う RAND()　　[0,1)の一様乱数
    2. 関数を使う RANDBETWEEN(a,b)　　[a,b]の整数一様乱数
    3. 「データ」－「データ分析」ツールの「乱数発生」を利用
       各種分布に従った乱数を生成できる

➢ 補足：Microsoft365 の新関数

- RANDARRAY() … 乱数を配列[行列]で生成（※配列数式で入力）
- 書式：= RANDARRAY(行数, 列数 [, 最小値] [, 最大値] [, 乱数の種類])
  - 1行1列目に数式を記入
  - 行数×列数のサイズで，最小値～最大値の範囲の指定乱数を生成
  - 最小値省略時0，最大値省略時1
  - 乱数の種類は，TRUEなら整数，FALSE(or省略)なら有理数（小数）
  - 記入後[Enter]のみで，行数×列数を自動記入し計算

【演習】 乱数で適当なLPを生成し Excel Solver で求解せよ

参考文献

1. *A. Schrijver: Theory of Linear and Integer Programming, John Wiley and Sons, 1986.*
2. *L.A. Wolsey: Integer Programming, John Wiley and Sons, 1998.*
3. *M. Conforti, G. Cornuejols and G.Zambelli: Integer Programming, Springer, 2014.*
4. 久保幹雄, J.P.ペドロソ, 村松正和, A.レイス：あたらしい数理最適化, 近代科学社,2012.
5. 久保幹雄, 小林和博, 斉藤努, 並木誠, 橋本英樹：Python言語によるビジネスアナリティクス, 近代科学社, 2016.
6. 藤澤克樹, 後藤順哉, 安井雄一郎：Excelで学ぶOR, オーム社, 2011.
7. 堀田敬介：えくせるであそぶ, 創成社, 2005.