問題解決

組合せ最適化と整数計画法
数独 sudoku

堀田 敬介

数独を解く

[例題出典] 斉藤努「データ分析ライブラリーを用いた最適化モデルの作り方」近代科学社 (2018) p.96 図7.3

- 数独（例1）
  - 3×3のブロックを9個並べた正方形升の各枠に1～9の数値を入れるパズル
    - ① 1～9を各行（全9行）に1つずつ入れる
    - ② 1～9を各列（全9列）に1つずつ入れる
    - ③ 1～9を各3×3のブロック（全9個）に1つずつ入れる

	1	2	3	4	5	6	7	8	9
1			6						1
2		7			6			5	
3	8			1		3	2		
4			5		4		8		
5		4		7		2		9	
6			8		1		7		
7			1	2		5			3
8		6			7			8	
9	2						4		

- 最適化問題の定式化（ベタ・Σ表記）
  - 0-1変数 $x_{ijk}=1$ … (i,j) に k が入る
  - 0-1変数 $x_{ijk}=0$ … (i,j) に k が入らない

81本
$x_{111}+x_{112}+x_{113}+x_{114}+x_{115}+x_{116}+x_{117}+x_{118}+x_{119}=1$ (1,1)←1～9
…
$x_{991}+x_{992}+x_{993}+x_{994}+x_{995}+x_{996}+x_{997}+x_{998}+x_{999}=1$ (9,9)←1～9

81本
$x_{111}+x_{121}+x_{131}+x_{141}+x_{151}+x_{161}+x_{171}+x_{181}+x_{191}=1$ 1行目←1
…
$x_{919}+x_{929}+x_{939}+x_{949}+x_{959}+x_{969}+x_{979}+x_{989}+x_{999}=1$ 9行目←9

81本
$x_{111}+x_{211}+x_{311}+x_{411}+x_{511}+x_{611}+x_{711}+x_{811}+x_{911}=1$ 1列目←1
…
$x_{199}+x_{299}+x_{399}+x_{499}+x_{599}+x_{699}+x_{799}+x_{899}+x_{999}=1$ 9列目←9

$x_{ijk}\in\{0,1\}$ $(i,j,k\in\{1,\ldots,9\})$ 9x9x9=729変数

81本（省略：3×3ブロック用制約）

$$\sum_{k=1}^{9} x_{ijk}=1(i,j=1,\ldots,9)$$

$$\sum_{j=1}^{9} x_{ijk}=1(i,k=1,\ldots,9)$$

$$\sum_{i=1}^{9} x_{ijk}=1(j,k=1,\ldots,9)$$

$$x_{ijk}\in\{0,1\}(i,j,k=1,\ldots,9)$$

81本（省略：3×3ブロック用制約）

数独を解く

- 数独の問題例を数式に
  - Excelを利用して数式を書く

$x_{136}=1$ (1,3)←6

CPLEX の数式で制約を書く

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1		1	2	3	4	5	6	7	8	9						↓cplexに貼り付ける式		
2	1			6						1		1	1			x[1,3,6] == 1;	x[1,9,1] == 1;	x[2,2,7]
3	2		7			6			5			1	2					
4	3	8			1		3	2				1	3	6	x[1,3,6] == 1;			
5	4			5		4		8				1	4		x[1,3,6] == 1;			
6	5		4		7		2		9			1	5		x[1,3,6] == 1;			
7	6			8		1		7				1	6		x[1,3,6] == 1;			
8	7			1	2		5			3		1	7		x[1,3,6] == 1;			
9	8		6			7			8			1	8		x[1,3,6] == 1;			
10	9	2						4				1	9	1	x[1,3,6] == 1; x[1,9,1] == 1;			
11												2	1		x[1,3,6] == 1; x[1,9,1] == 1;			
12												2	2	7	x[1,3,6] == 1; x[1,9,1] == 1; x[2,2,7] == 1;			
13												2	3		x[1,3,6] == 1; x[1,9,1] == 1; x[2,2,7] == 1;			
14												2	4		x[1,3,6] == 1; x[1,9,1] == 1; x[2,2,7] == 1;			
15												2	5	6	x[1,3,6] == 1; x[1,9,1] == 1; x[2,2,7] == 1; x[2,5,6]			
16												2	6		x[1,3,6] == 1; x[1,9,1] == 1; x[2,2,7] == 1; x[2,5,6]			

数独をCPLEXで解く

- 新規プロジェクトの作成
  - ① [ファイル(F)]ー[新規(N)]ー[OPLプロジェクト]を選択
  - ② [プロジェクト名] を記入（例：Sudoku）し，2カ所にチェックする
    - ☑ デフォルトの実行構成の追加
    - ☑ モデルの作成
    - ~~☑ データの作成~~
  - ③ [終了]をクリック

プロジェクト名は自由だが，半角英数で何の問題を解こうとしているのかが分かる名前が良い

- プロジェクト内のいくつかの名前を変更
  - ✓ [構成1] → [config1]　※日本語を英語に変更しないと実行時エラーになる
  - ✓ モデルファイル [Sudoku.mod] → [np.mod]
  - ✓ データファイルは使わない

- モデルファイル~~・データファイル~~を記述し保存（次ページ参照）
- [config1]にモデルファイル~~とデータファイル~~をセットし，解く

数独を解く

- モデルファイル（np.mod）の中身の記述

```
range I = 1..9;// 行の添え字の範囲 [1..9]を指定
range J = 1..9;// 列の添え字の範囲 [1..9]を指定
range K = 1..9;// ナンプレ枠内に入る数値(1～9)

dvar int+ x[I,J,K] in 0..1;// 変数宣言:0-1変数ベクトル(size:IxJxK)

maximize
  sum(i in I) x[i,1,1];// ダミー（この和は常に1）
subject to {
  forall(i in I) {
    forall(j in J) {
      sum(k in K) x[i,j,k] == 1;// 1枡に入る数値は1つ
    };
  };
  forall(i in I) {
    forall(k in K) {
      sum(j in J) x[i,j,k] == 1;// 1行の各枡に入る数値(1-9)は1つずつ
    };
  };
  forall(j in J) {
    forall(k in K) {
      sum(i in I) x[i,j,k] == 1;// 1列の各枡に入る数値(1-9)は1つずつ
    };
  };
  forall(k in K) {// 1つの3x3枠内の各枡に入る数値(1-9)は1つずつ
    sum(i in 1..3) sum(j in 1..3) x[i,j,k] == 1;
    sum(i in 1..3) sum(j in 4..6) x[i,j,k] == 1;
    sum(i in 1..3) sum(j in 7..9) x[i,j,k] == 1;
    sum(i in 4..6) sum(j in 1..3) x[i,j,k] == 1;
    sum(i in 4..6) sum(j in 4..6) x[i,j,k] == 1;
    sum(i in 4..6) sum(j in 7..9) x[i,j,k] == 1;
    sum(i in 7..9) sum(j in 1..3) x[i,j,k] == 1;
    sum(i in 7..9) sum(j in 4..6) x[i,j,k] == 1;
    sum(i in 7..9) sum(j in 7..9) x[i,j,k] == 1;
  };
※ここに記入（改行は適当に）
};
```

この目的関数に意味はない

81本

81本

81本

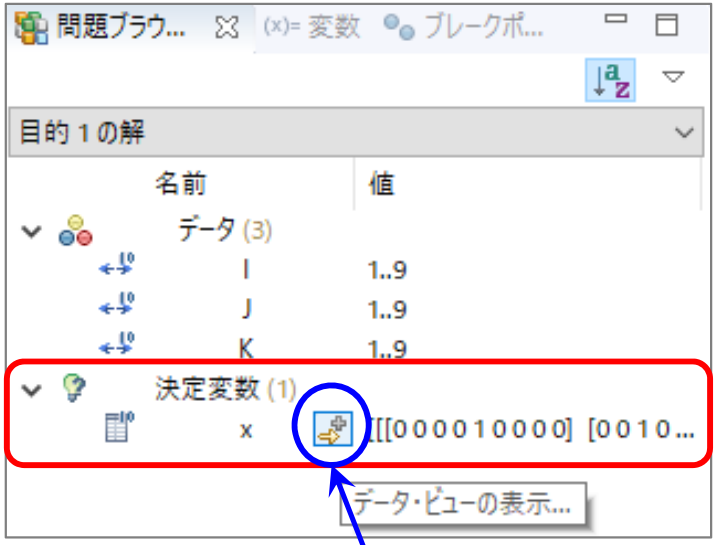
81本

問題（ex1）で既に与えられている数値をモデルファイルの最後に挿入する

```
x[1,3,6] == 1; x[1,9,1] == 1;
x[2,2,7] == 1; x[2,5,6] == 1;
x[2,8,5] == 1; x[3,1,8] == 1;
x[3,4,1] == 1; x[3,6,3] == 1;
x[3,7,2] == 1; x[4,3,5] == 1;
x[4,5,4] == 1; x[4,7,8] == 1;
x[5,2,4] == 1; x[5,4,7] == 1;
x[5,6,2] == 1; x[5,8,9] == 1;
x[6,3,8] == 1; x[6,5,1] == 1;
x[6,7,7] == 1; x[7,3,1] == 1;
x[7,4,2] == 1; x[7,6,5] == 1;
x[7,9,3] == 1; x[8,2,6] == 1;
x[8,5,7] == 1; x[8,8,8] == 1;
x[9,1,2] == 1; x[9,7,4] == 1;
```

数独をCPLEXで解く

➢ 計算結果の確認



このボタン押し，データ・ビュー表示

np.mod | x の値

I (サイズ 9)	J (サイズ 9)	K (サイズ 9)	値
1	1	1	0
1	2	1	0
1	3	1	0
1	4	1	0
1	5	1	0
1	6	1	0
1	7	1	0
1	8	1	0
1	9	1	1
2	1	1	1
2	2	1	0
2	3	1	0

$x_{191}=1$ (1,9)←1
$x_{211}=1$ (2,1)←1

問題 | スクリプト... | 解 | 競合 | 統計情報 | プロファ...

最適値[1]には意味は無い

```
// solution (optimal) with objective 1
// Quality Incumbent solution:
// MILP objective                                 1.0000000000e+00
// MILP solution norm |x| (Total, Max)            8.10000e+01  1.00000e+00
// MILP solution error (Ax=b) (Total, Max)        0.00000e+00  0.00000e+00
// MILP x bound error (Total, Max)                0.00000e+00  0.00000e+00
// MILP x integrality error (Total, Max)          0.00000e+00  0.00000e+00
// MILP slack bound error (Total, Max)            0.00000e+00  0.00000e+00
//

x = [[[0 0 0 1 0 0 0 0 0]
      [0 0 1 0 0 0 0 0 0]
      [0 0 0 0 0 1 0 0 0]
      [0 0 0 0 0 0 0 1 0]
      [0 1 0 0 0 0 0 0 0]
      [0 0 0 0 0 0 1 0 0]
      [0 0 0 0 0 0 0 0 1]
      [0 0 0 1 0 0 0 0 0]
      [1 0 0 0 0 0 0 0 0]]
     [[1 0 0 0 0 0 0 0 0]
      [0 0 0 0 0 0 1 0 0]
      [0 1 0 0 0 0 0 0 0]
      [0 0 0 0 0 0 0 0 1]
      [0 0 0 0 0 1 0 0 0]
      [0 0 0 1 0 0 0 0 0]
      [0 0 1 0 0 0 0 0 0]
      [0 0 0 0 1 0 0 0 0]
      [0 0 0 0 0 0 0 1 0]]
     [[0 0 0 0 0 0 0 1 0]
      [0 0 0 0 0 0 0 0 1]
      [0 0 0 1 0 0 0 0 0]
      [1 0 0 0 0 0 0 0 0]
      [0 0 0 0 1 0 0 0 0]
      [0 0 1 0 0 0 0 0 0]
      [0 1 0 0 0 0 0 0 0]
      [0 0 0 0 0 1 0 0 0]
      [0 0 0 0 0 0 1 0 0]]
     [0 0 0 0 0 0 1 0 0]
      [1 0 0 0 0 0 0 0 0]
      [0 0 0 0 1 0 0 0 0]
      [0 0 1 0 0 0 0 0 0]
      [0 0 0 1 0 0 0 0 0]
      [0 0 0 0 0 0 0 0 1]
```

[解]タブの中身

最適解729変数
（=[9×9]×9個）

数独をCPLEXで解く

➢ 計算結果の処理

①このボタンを[1回]押す　→この値の[昇順]にソートされる

np.mod　x の値

I (サイズ 9)	J (サイズ 9)	K (サイズ 9)	値
1	1	1	0
1	2	1	0
1	3	1	0
1	4	1	0
1	5	1	0
1	6	1	0
1	7	1	0
1	8	1	0
1	9	1	1
2	1	1	1
2	2	1	0

np.mod　x の値

I (サイズ 9)	J (サイズ 9)	K (サイズ 9)	値
9	9	4	0
9	9	5	0
9	9	6	0
9	9	7	0
9	9	8	0
1	1	5	1
1	2	3	1
1	3	6	1
1	4	8	1
1	5	2	1
1	6	7	1
1	7	9	1
1	8	4	1
1	9	1	1
2	1	1	1

②下にスクロールさせて，[値]に[1]が出てくる最初の行を表示させる
③その行を（どこでも良いので）クリックする
（※背景色が反転する）

⑥反転上で右クリックし，[クリップボードにコピー]選択

np.mod　x の値

I (サイズ 9)	J (サイズ 9)	K (サイズ 9)	値
8	8	8	1
8	9	2	1
9	1	2	1
9	2	5	1
9	3	7	1
9	4	6	1
9			
9			
9	7	4	1
9	8	1	1
9	9	9	1

クリップボードにコピー

④さらに一番下までスクロールさせ最後の行を表示させる
⑤最後の行を[Shift]キーを押しながらクリックする（範囲選択）
（③～④の全背景色が反転）

np.mod　x の値

I (サイズ 9)	J (サイズ 9)	K (サイズ 9)	値
8	8	8	1
8	9	2	1
9	1	2	1
9	2	5	1
9	3	7	1
9	4	6	1
9	5	3	1
9	6	8	1
9	7	4	1
9	8	1	1
9	9	9	1

問題　スクリプト...　解　競合　緩和

数独をgurobiで解く(1)

- cplexの「モデルファイル(*.mod)」と「データファイル(*.dat)」を使って「lpファイル(*.lp)」を生成する
  - 例） モデルファイル [np.mod], データファイル [npex1.dat]
    → 生成する lpファイル [npex1.lp]
  - [Win]+[R] キー で [ファイル名を指定して実行] d-boxを起動する
    - 枠内で cmd [Enter]
  - コマンドプロンプト command prompt のウィンドウ(黒い画面)が起動する
  - 以降，コマンドプロンプト内でコマンド(命令文)を打って順次命令を実行する

(1) モデルファイルとデータファイルがあるフォルダに移動する

```
cd [フォルダへのパス]  [Enter]
```

(2) 以下のコマンドを実行する

```
oplrun -e npex1.lp np.mod npex1.dat  [Enter]
```

  - この結果，モデルファイル [np.mod] とデータファイル [npex1.dat] と同じフォルダ内に，lpファイル [npex1.lp] が出来る(※確認すること)

数独をgurobiで解く(1)

- gurobi を起動して問題を解き，最適解を得る
  - コマンドプロンプトで，以下の命令文を打って gurobi を起動する

```
gurobi [Enter]
```

  - 起動した gurobi 内で，順次，以下の命令文を打って問題を解いていく

(1) 問題を記述してある lpファイル(npex1.lp)を読み込み，model へセット

```
model = read("npex1.lp") [Enter]
```

(2) 解く(最適化計算を開始する) ※読込に失敗しているとエラーとなる

```
model.optimize() [Enter]
```

(3) 最適解を表示する ※最適解が求まっていない場合はエラーとなる

```
model.printAttr('X') [Enter]
```

(4) 最適値(目的関数値)を表示する ※同上

```
model.ObjVal [Enter]
```

(5) 最適解をファイル(*.sol)に出力する ※ファイル名は好きに

```
model.write("npex1.sol") [Enter]
```

数独をgurobiで解く(1)

- gurobi のその他，知っておくと便利な命令文
  - いずれも gurobi を起動して，gurobi内で実行する

    (a) ヘルプを表示する

```
help()   [Enter]
```

    (b) 全ての最適解(値が0の解)を表示する

```
for v in model.getVar() :    [Enter]
    print( v.VarName, ":", v.X)   [Enter]
```

  - 最適解を表示する命令文「m.printAttr('X')」は，値が0となる解は表示しない
  - 2行目の print 文は，必ず字下げ(インデント)して書くこと(Pythonの文法)
  - 字下げは[Tab]キーを使うと良い(※面倒でなければ，半角スペースでも可)
  - model.getVar() でモデルから変数Var(variableの頭3文字) を get する命令
  - get した各変数をインデックス v として，for文で繰り返す(2行目を繰り返す)
  - v.VarName は，ゲットした各変数の「名称」を意味する予約語
  - v.X は，ゲットした各変数の「値」を意味する予約語
  - 以上より，各変数を1つずつ「名称 : 値」の形で画面に表示(print)する

数独をgurobiで解く(

➢ 問題(ex1)を python & guro

```
# coding: Shift_JIS
from gurobipy import *
```
①

1つのファイル「np.py」に
①②③の順に記述して保存

```
mod.addConstr(x[1,3,6]==1)
mod.addConstr(x[1,9,1]==1)
mod.addConstr(x[2,2,7]==1)
…(中略)…
mod.addConstr(x[9,7,4]==1)
```

```
# ####### 実行 #######
if __name__=="__main__":
    mod = np()                          # モデルの生成
    mod.write("npex1.lp")               # lpファイルを出
    mod.optimize()                      # 最適化実行
    print("¥n optimal value = ", mod.ObjVal)      # 最
    mod.printAttr('X')                  # 最適解の表示
    mod.write("npex1.sol")              # 最適解をsolフ
```
③

```
# ####### 定式化 #######
def np():
    mod = Model("sudoku")

    # 変数設定
    x = {}
    for i in range(1,10):
      for j in range(1,10):
        for k in range(1,10):
            x[i,j,k] = mod.addVar(vtype="B", name="x(%s,%s,%s)" % (i,j,k))
    mod.update()

    # 制約条件の設定
    for i in range(1,10):
      for j in range(1,10):
        mod.addConstr(quicksum(x[i,j,k] for k in I) == 1)
      for k in range(1,10):
        mod.addConstr(quicksum(x[i,j,k] for j in I) == 1)
    for j in range(1,10):
      for k in range(1,10):
        mod.addConstr(quicksum(x[i,j,k] for i in I) == 1)
    for k in range(1,10):
      mod.addConstr(quicksum(x[i,j,k] for i in [1,2,3] for j in [1,2,3]) == 1)
      mod.addConstr(quicksum(x[i,j,k] for i in [1,2,3] for j in [4,5,6]) == 1)
      mod.addConstr(quicksum(x[i,j,k] for i in [1,2,3] for j in [7,8,9]) == 1)
      mod.addConstr(quicksum(x[i,j,k] for i in [4,5,6] for j in [1,2,3]) == 1)
      mod.addConstr(quicksum(x[i,j,k] for i in [4,5,6] for j in [4,5,6]) == 1)
      mod.addConstr(quicksum(x[i,j,k] for i in [4,5,6] for j in [7,8,9]) == 1)
      mod.addConstr(quicksum(x[i,j,k] for i in [7,8,9] for j in [1,2,3]) == 1)
      mod.addConstr(quicksum(x[i,j,k] for i in [7,8,9] for j in [4,5,6]) == 1)
      mod.addConstr(quicksum(x[i,j,k] for i in [7,8,9] for j in [7,8,9]) == 1)
※ここに問題例の式を記入
    # 目的関数の設定
    mod.setObjective(quicksum(x[i,1,1] for i in range(1,10)), GRB.MAXIMIZE)
    mod.update()
    mod.__data = x
    return mod
```
②

数独をgurobiで解く(2)

- Pythonファイル(np.py)をgurobi上で実行し，解く
  - [Win]+[R] キー で [ファイル名を指定して実行] d-boxを起動する
    - 枠内で cmd [Enter]
  - コマンドプロンプト command prompt のウィンドウ(黒い画面)が起動する
  - コマンドプロンプト内でコマンド(命令文)を打って順次命令を実行する

(1) 実行ファイルがあるフォルダに移動する

```
cd [フォルダへのパス] [Enter]
```

(2) 以下の命令文を打って gurobi を起動する

```
gurobi [Enter]
```

  - 起動した gurobi 内で，以下の命令文を打って問題を解く

```
gurobi> exec( open("np.py").read() ) [Enter]
```

※python3系の場合

※python2系の場合の命令文は以下

```
gurobi> execfile("np.py") [Enter]
```

数独をgurobiで解く(2)

- 実行結果

```
gurobi> exec(open("np.py").read())
Gurobi Optimizer version 9.5.2 build v9.5.2rc0 (win64)
Thread count: 10 physical cores, 20 logical processors, using up to 20 threads
Optimize a model with 352 rows, 729 columns and 2944 nonzeros
Model fingerprint: 0xfcb0ef05
Variable types: 0 continuous, 729 integer (729 binary)
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [1e+00, 1e+00]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 1e+00]
Presolve removed 352 rows and 729 columns
Presolve time: 0.00s
Presolve: All rows and columns removed

Explored 0 nodes (0 simplex iterations) in 0.00 seconds (0.00 work units)
Thread count was 1 (of 20 available processors)

Solution count 1: 1

Optimal solution found (tolerance 1.00e-04)
Best objective 1.000000000000e+00, best bound 1.000000000000e+00, gap 0.0000%

optimal value =  1.0

    Variable            X
-------------------------
    x(1,1,5)            1
    x(1,2,3)            1
    x(1,3,6)            1
    x(1,4,8)            1
    x(1,5,2)            1
    x(1,6,7)            1
    x(1,7,9)            1
    x(1,8,4)            1
    x(1,9,1)            1
    x(2,1,1)            1
```

[例題出典] wikipedia「数独」問題例

数独を解く

- 数独解く（例2）
  - 3×3のブロックを9個並べた正方形升の各枠に1～9の数値を入れるパズル
    - 1～9を各行（全9行）に1つずつ入れる
    - 1～9を各列（全9列）に1つずつ入れる
    - 1～9を各3×3のブロック（全9個）に1つずつ入れる

	1	2	3	4	5	6	7	8	9
1	5	3			7				
2	6			1	9	5			
3		9	8					6	
4	8				6				3
5	4			8		3			1
6	7				2				6
7		6					2	8	
8				4	1	9			5
9					8			7	9

数独を解く

- 数独を解く（例3）
  - 3×3のブロックを9個並べた正方形升の各枠に1～9の数値を入れるパズル
    - 1～9を各行（全9行）に1つずつ入れる
    - 1～9を各列（全9列）に1つずつ入れる
    - 1～9を各3×3のブロック（全9個）に1つずつ入れる
  - 例題を1つ（見つけて）Excelファイルのex3シートに記入する
    - 答えがちゃんとある問題をつくること（適当に数値を入れると答えがない問題になる）
  - CPLEXの問題を表す制約をこのシートで計算した数式に入れ替えて解く