

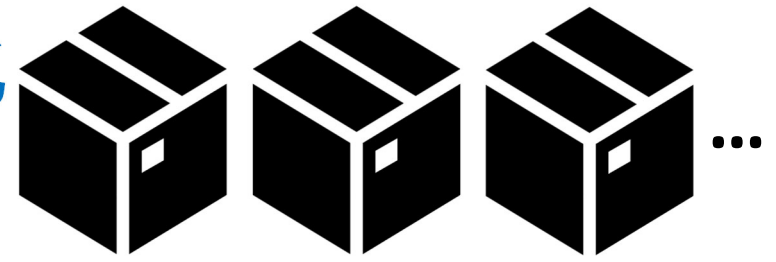
問題解決のための最適化

組合せ最適化と整数計画法

2. ビンパッキング問題

堀田 敬介

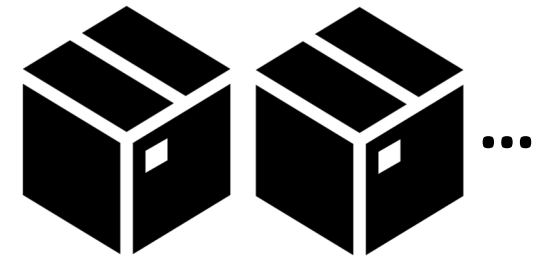
ビンパッキング問題の最適化



➤ ビンパッキング問題 bin packing problem

- 容量 B のビン bin が(十分な数)ある
- アイテムが n 個あり, それぞれサイズがある
- 全てのアイテムをビン bin に詰める packing とき, 必要なビンの最小数を求めたい. どう詰めたらビンが少なくて済むか?
- 暗黙の仮定として, ビンより大きなアイテムはない, とする(あったら解けない)
- 自明な解として, ビンをアイテム数と同じ n 個 用意すれば, 全アイテムをビンに収納可能である. つまり, 1つのビンに1つのアイテムを詰めればよい. 故に, 問題では, もっと少ないビン数の答えが欲しい

ビンパッキング問題の最適化



➤ 最適化問題の定式化(変数設定・係数表記)

➤ 0-1変数 $x_{ij} = \begin{cases} 1 & \dots \text{アイテム} i \text{ をビン} j \text{ に詰める} \\ 0 & \dots \text{アイテム} i \text{ をビン} j \text{ に詰めない} \end{cases}$

➤ 0-1変数 $y_j = \begin{cases} 1 & \dots \text{ビン} j \text{ を使う} \\ 0 & \dots \text{ビン} j \text{ を使わない} \end{cases}$

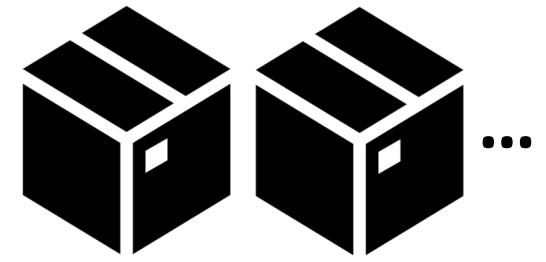
➤ アイテムの集合 $I = \{1, \dots, n\}$

➤ ビンの集合 $J = \{1, \dots, m\}$ ※ $m \leq n$

ビン数はアイテム数以下

➤ アイテム i のサイズを s_i とする ($\mathbf{s} = (s_1, \dots, s_n)$)

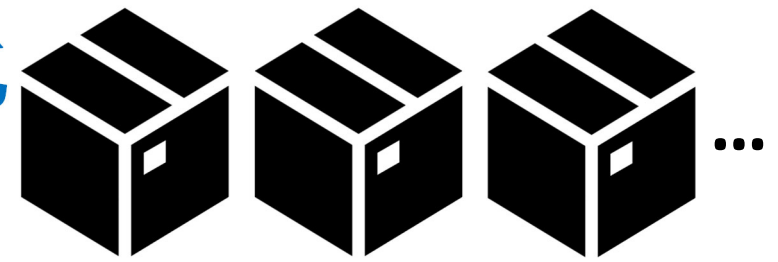
ビンパッキング問題の最適化



➤ 最適化問題の定式化(ベタ表記 $\Leftrightarrow$ Σ 表記)

$\text{min. } y_1 + y_2 + \dots + y_m$	$\longleftrightarrow$	$\text{min. } \sum_{j=1}^m y_j$
$\text{s. t. } x_{11} + x_{12} + \dots + x_{1m} = 1$ $\dots$ $x_{n1} + x_{n2} + \dots + x_{nm} = 1$	$\longleftrightarrow$	s. t. $\sum_{j=1}^m x_{ij} = 1 \quad (i \in I)$
$s_1 x_{11} + s_2 x_{21} + \dots + s_n x_{n1} \leq B y_1$ $\dots$ $s_1 x_{1m} + s_2 x_{2m} + \dots + s_n x_{nm} \leq B y_m$	$\longleftrightarrow$	$\sum_{i=1}^n s_i x_{ij} \leq B y_j \quad (j \in J)$
$x_{11}, x_{12}, \dots, x_{nm} \in \{0, 1\}$ $y_1, y_2, \dots, y_m \in \{0, 1\}$	$\longleftrightarrow$ $\longleftrightarrow$	$x_{ij} \in \{0, 1\} \quad (i \in I, j \in J)$ $y_j \in \{0, 1\} \quad (j \in J)$

ビンパッキング問題の最適化



➤ ビンパッキング問題 bin packing problem

- 容量 B のビン bin が(十分な数)ある
- アイテムが n 個あり, それぞれサイズがある
- 全てのアイテムをビン bin に詰める packing とき, 必要なビンの最小数を求めたい. どう詰めたらビンが少なくて済むか?
- 暗黙の仮定として, ビンより大きなアイテムはない, とする
- 1つのビンに1つのアイテムを詰めるが自明解(必要ビン数=アイテム数)

➤ ビンパッキング問題のモデル化例(ex1)

- 容量 $B=30$ のビン bin が(十分な数)ある
- アイテムが $n=15$ 個あり, 各アイテムのサイズは下表の通り
- 必要最小数のビン数を求めよ

$$\begin{cases} n = 15 \\ m = 15 \\ B = 30 \end{cases}$$

品物	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
サイズ	15	2	4	7	8	9	12	6	4	5	3	2	7	9	8

ビンパッキング問題をCPLEXで解く

➤ 新規プロジェクトの作成

- ① [ファイル(F)]－[新規(N)]－[OPLプロジェクト]を選択
- ② [プロジェクト名] を記入 (例: **BinPacking**) し, 3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 [BinPacking.mod] → [bp.mod]
- ✓ データファイル [BinPacking.dat] → [bpex1.dat]

➤ 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

ビンパッキング問題をCPLEXで解く

➤ モデルファイル(bp.mod)の中身の記述

```
int i_max = ...; // アイテム数
int j_max = ...; // ビンの最大数
int binB = ...; // ビンの容量

range I = 1..i_max;
range J = 1..j_max;

int s[I] = ...; // 各アイテムのサイズ

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

minimize
    sum(j in J) y[j]; // 使用するビンの数を最小にしたい
subject to {
    forall(i in I) {
        sum(j in J) x[i,j] == 1; // 各アイテムはどこかのビンに詰める
    }
    forall(j in J) {
        sum(i in I) s[i]*x[i,j] <= binB*y[j]; // 各ビンに詰めるアイテムのサイズ合計は容量binB以下
    }
}
```

ビンパッキング問題をCPLEXで解く

➤ データファイル (bpex1.dat) の中身の記述

```
i_max = 15; // アイテム数  
j_max = 15; // ビンの最大数    ※j_max ≤ i_max  
binB = 30; // (1つの)ビンの容量  
s = [15,2,4,7,8,9,12,6,4,5,3,2,7,9,8];
```

品物	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
サイズ	15	2	4	7	8	9	12	6	4	5	3	2	7	9	8

ビンパッキング問題をCPLEXで解く

➤ 結果([解]タブ)

必要なビンの数(最小値)は4つ

最適値 = 4

```
// solution (optimal) with objective 4  
// Quality Incumbent solution:  
// MILP objective  
// MILP solution norm |x| (Total, Max)  
// MILP solution error (Ax=b) (Total, Max)  
// MILP x bound error (Total, Max)  
// MILP x integrality error (Total, Max)  
// MILP slack bound error (Total, Max)  
//  
y =  
x =
```

4.0000000000e+00

1.90000e+01 1.00000e+00

0.00000e+00 0.00000e+00

0.00000e+00 0.00000e+00

0.00000e+00 0.00000e+00

0.00000e+00 0.00000e+00

最適解

```
y = [1 1 1 1 0 0 0 0 0 0 0 0 0 0 0];  
x = [1 0 0 0 0 0 0 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 0 0 0 0 0 0 0 0 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 0 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 0 0 0 0 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 0 0 0 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 0 0 0 0 0 0 0 0 0]  
[1 0 0 0 0 0 0 0 0 0 0 0 0 0 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 0 0 0 0 0 0]  
[0 0 0 1 0 0 0 0 0 0 0 0 0 0 0];
```

ビン 1,2,3,4 の4つを使う

※ $y[j]$ はビン j を使うかどうかの0-1変数
※ $x[i,j]$ はアイテム i をビン j へ詰めるかどうかの0-1変数. 行がアイテム, 列がビンに該当

ビン1へ アイテム1,2,3,4,12 を詰める (サイズ30)

ビン2へ アイテム5,6,7 を詰める (同29)

ビン3へ アイテム8,9,10,11,13 を詰める (同25)

ビン4へ アイテム14,15 を詰める (同17)

品物	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
サイズ	15	2	4	7	8	9	12	6	4	5	3	2	7	9	8
詰め先	1	1	1	1	2	2	2	3	3	3	3	1	3	4	4

ビンパッキング問題を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 [bp.mod], データファイル [bpex1.dat]
→ 生成する lpファイル [bpex1.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
`oplrn -e bpex1.lp bp.mod bpex1.dat [Enter]`
- この結果, モデルファイル [bp.mod] とデータファイル [bpex1.dat] と同じフォルダ内に, lpファイル [bpex1.lp] が出来る (※確認すること)

ビンパッキング問題を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 (bpex1.lp)を読み込み, model へセット

```
model = read("bpex1.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("bpex1.sol") [Enter]
```

ビンパッキング問題を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

ビンパッキング問題をgurobiで解く(2)

➤ 問題(ex1)を python & gurobi で記述(bp.py)

coding: Shift_JIS

from gurobipy import *

①

例題設定

def make_data_ex1():

binB = 30

s = [15,2,4,7,8,9,12,6,4,5,3,2,7,9,8]

return binB,s

1つのファイル「bp.py」に
①②③の順に記述して保存

実行

if __name__=="__main__":

binB,s = make_data_ex1()

mod = bp(binB,s)

mod.write("bpex1.lp")

mod.optimize()

print("¥n optimal value = ", mod.ObjVal)

mod.printAttr('X')

mod.write("bpex1.sol")

③

データ

モデル

lpファイル

最適化

最適解の表示

最適解をsolファイルに出力

定式化

def bp(binB,s):

mod = Model("binpacking problem")

変数設定

x,y = {},{}

for j in range(len(s)):

y[j] = mod.addVar(vtype="B", name="y(%s)" % j)

for i in range(len(s)):

x[i,j] = mod.addVar(vtype="B", name="x(%s,%s)" % (i,j))

mod.update()

制約条件の設定

for i in range(len(s)):

mod.addConstr(quicksum(x[i,j] for j in range(len(s))) == 1)

for j in range(len(s)):

mod.addConstr(quicksum(s[i]*x[i,j] for i in range(len(s))) <= binB*y[j])

目的関数の設定

mod.setObjective(quicksum(y[j] for j in range(len(s))), GRB.MINIMIZE)

mod.update()

mod.__data = x,y

return mod

②

ビンパッキング問題をgurobiで解く(2)

➤ Pythonファイル(bp.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("bp.py").read() ) [Enter]
```

※python3系の場合

※python2系の場合の命令文は以下

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

ビンパッキング問題をgurobiで解く(2)

実行結果

```

optimal value = 4.0

Variable      X
-----
y(0)          1
x(1,0)        1
x(6,0)        1
x(7,0)        1
x(8,0)        1
y(8)          1
x(0,8)        1
x(11,8)       1
y(12)         1
x(3,12)       1
x(5,12)       1
x(9,12)       1
x(13,12)      1
y(13)         1
x(2,13)       1
x(4,13)       1
x(10,13)      1
x(12,13)      1
x(14,13)      1

```

```

gurobi> exec(open("bp.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 30 rows, 240 columns and 465 nonzeros
Model fingerprint: 0x8d28ac87
Variable types: 0 continuous, 240 integer (240 binary)
Coefficient statistics:
  Matrix range      [1e+00, 3e+01]
  Objective range   [1e+00, 1e+00]
  Bounds range      [1e+00, 1e+00]
  RHS range         [1e+00, 1e+00]
Found heuristic solution: objective 8.0000000
Presolve time: 0.00s
Presolved: 30 rows, 240 columns, 465 nonzeros
Variable types: 0 continuous, 240 integer (240 binary)

Root relaxation: objective 3.36667e+00, 55 iterations, 0.00 seconds (0.00 work units)

   Nodes      |   Current Node   |   Objective Bounds   |         Work
  Expl Unexpl |  Obj  Depth IntInf | Incumbent  BestBd   Gap   | It/Node Time
-----
    0     0    3.36667    0   8      8.00000    3.36667  57.9%   -    0s
H    0     0          5.00000    3.36667  32.7%   -    0s
H    0     0          4.00000    3.36667  15.8%   -    0s
    0     0    3.36667    0   8      4.00000    3.36667  15.8%   -    0s

Explored 1 nodes (55 simplex iterations) in 0.02 seconds (0.00 work units)
Thread count was 20 (of 20 available processors)

Solution count 3: 4 5 8

Optimal solution found (tolerance 1.00e-04)
Best objective 4.000000000000e+00, best bound 4.000000000000e+00, gap 0.0000%

```

Bin数=4本

Bin1←item2,7,8,9 [size2+12+6+4=24]

Bin7←item1,12 [size15+2=17]

Bin13←item4,6,10,14 [size7+9+5+9=30]

Bin14←item3,5,11,13,15 [size4+8+3+7+8=30]

品物	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
サイズ	15	2	4	7	8	9	12	6	4	5	3	2	7	9	8
詰め先	1	1	1	1	2	2	2	3	3	3	3	1	3	4	4

ビンパッキング問題を解く

$s = [9, 1, 7, 14, 1, 15, 4, 3, 14, 13, 11, 8, 2, 13, 3, 3, 4, 15, 6, 2, 4, 4, 8, 14, 2, 8, 7, 14, 5, 1, 12, 13, 15, 6, 9, 1, 14, 11, 11, 6, 9, 3, 7, 12, 6, 12, 14, 12, 4, 12]$

➤ ビンパッキング問題の最適化(例2)

- 容量 $B = 60$ のビンが, 十分な数ある ($m = 10$ 個とする)
- アイテムが $n = 50$ 個あり, 各サイズは右上の通り
- 必要なビンの最小数を求めたい. どう詰めたらビンが最小になるか?

➤ 最適化問題の(強化した)定式化(ベタ表記 $\leftrightarrow$ Σ 表記)

$\min. y_1 + y_2 + \dots + y_{10}$	$\longleftrightarrow$	$\min. \sum_{j=1}^m y_j$	$\begin{cases} n = 50 \\ m = 10 \\ B = 60 \end{cases}$
$\text{s. t. } x_{11} + x_{12} + \dots + x_{1,10} = 1$	}	s. t.	
$\dots$		$\sum_{j=1}^m x_{ij} = 1 \quad (i \in I)$	
$x_{50,1} + x_{50,2} + \dots + x_{50,10} = 1$	}	$\sum_{i=1}^n s_i x_{ij} \leq B y_j \quad (j \in J)$	
$s_1 x_{11} + s_2 x_{21} + \dots + s_{50} x_{50,1} \leq B y_1$			
$\dots$	}	$x_{ij} \in \{0, 1\} \quad (i \in I, j \in J)$	
$s_1 x_{1,10} + s_2 x_{2,10} + \dots + s_{50} x_{50,10} \leq B y_{10}$		$y_j \in \{0, 1\} \quad (j \in J)$	
$x_{11}, x_{12}, \dots, x_{50,10} \in \{0, 1\}$	}	$x_{ij} \leq y_j \quad (i \in I, j \in J)$	定式化強化のための追加式
$y_1, y_2, \dots, y_{10} \in \{0, 1\}$		$y_j \geq y_{j+1} \quad (j \in \{1, \dots, m-1\})$	
$x_{11} \leq y_1, x_{21} \leq y_1, \dots, x_{50,1} \leq y_1$	}		
$\dots$			
$x_{1,10} \leq y_{10}, x_{2,10} \leq y_{10}, \dots, x_{50,10} \leq y_{10}$	}		
$y_1 \geq y_2 \geq \dots \geq y_{10}$			

ビンパッキング問題を解く

➤ 例2(ex2) 結果 ([解]タブ)

必要なビンの数(最小値)は9つ

```
// solution (optimal) with objective 9
// Quality Incumbent solution:
// MILP objective
// MILP solution norm |x| (Total, Max)
// MILP solution error (Ax=b) (Total, Max)
// MILP x bound error (Total, Max)
// MILP x integrality error (Total, Max)
// MILP slack bound error (Total, Max)
//
```

最適値 = 9

9.0000000000e+00
5.90000e+01 1.00000e+00
0.00000e+00 0.00000e+00
0.00000e+00 0.00000e+00
0.00000e+00 0.00000e+00
0.00000e+00 0.00000e+00

```
y = [1 1 1 1 1 1 1 1 0];
x = [1 0 0 0 0 0 0 0 0 0]
```

← ビン 1,...,9 の9つを使う

※ $y[j]$ はビン j を使うかどうかの0-1変数
※ $x[i,j]$ はアイテム i をビン j へ詰めるかどうかの0-1変数. 行がアイテム, 列がビンに該当

最適解

ビンパッキング問題をgurobiで解く(2)

➤ 問題(ex2)をpython &

coding: Shift_JIS

from gurobipy import *

①

例題設定

def make_data_ex2():

 binB = 50

 s = [9,1,7,14,1,15,4,3,14,13,11,8,2,
13,3,3,4,15,6,2,4,4,8,14,2,8,7,14,5,1,12,
13,15,6,9,1,14,11,11,6,9,3,7,12,6,12,14,
12,4,12]

 return binB,s

1つのファイル「bp-str.py」に
①②③の順に記述して保存

実行

if __name__=="__main__":

 binB,s = make_data_ex2()

 mod = bp(binB,s)

 mod.write("bpex2.lp")

 mod.optimize()

 print("¥n optimal value = ", mod.ObjVal)

 mod.printAttr('X')

 mod.write("bpex2.sol")

③

データ
モデル
lpファイル
最適化
最適値
最適解
最適解

定式化

def bp(binB,s):

 mod = Model("binpacking problem")

変数設定

 x,y = {},{}
 for j in range(len(s)):

 y[j] = mod.addVar(vtype="B", name="y(%s)" % j)

 for i in range(len(s)):

 x[i,j] = mod.addVar(vtype="B", name="x(%s,%s)" % (i,j))

 mod.update()

制約条件の設定

 for i in range(len(s)):

 mod.addConstr(quicksum(x[i,j] for j in range(len(s))) == 1)

 for j in range(len(s)):

 mod.addConstr(quicksum(s[i]*x[i,j] for i in range(len(s))) <= binB*y[j])

 for j in range(len(s)):

 for i in range(len(s)):

 mod.addConstr(x[i,j] <= y[j])

 for j in range(len(s)-1):

 mod.addConstr(y[j] >= y[j+1])

定式化強化の
ための追加式

目的関数の設定

 mod.setObjective(quicksum(y[j] for j in range(len(s))), GRB.MINIMIZE)

 mod.update()

 mod.__data = x,y

 return mod

ビンパッキング問題をgurobiで解く(2)

実行結果

```
gurobi> exec(open("bp-str.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 2649 rows, 2550 columns and 10148 nonzeros
Model fingerprint: 0x5f8a2fa6
Variable types: 0 continuous, 2550 integer (2550 binary)
Coefficient statistics:
  Matrix range      [1e+00, 5e+01]
  Objective range   [1e+00, 1e+00]
  Bounds range      [1e+00, 1e+00]
  RHS range         [1e+00, 1e+00]
Found heuristic solution: objective 39.0000000
Presolve removed 674 rows and 563 columns
Presolve time: 0.06s
Presolved: 1975 rows, 1987 columns, 7709 nonzeros
Variable types: 0 continuous, 1987 integer (1987 binary)

Root relaxation: objective 8.080000e+00, 2388 iterations, 0.06 seconds (0.13 work units)

   Nodes      |   Current Node   |   Objective Bounds   |         Work
  Expl Unexpl |  Obj  Depth IntInf | Incumbent  BestBd   Gap   | It/Node Time
-----
H    0     0    9.00000    0   2    39.00000    9.00000  76.9%   -    0s
   0     0    9.00000    0   2    9.0000000    9.00000  0.00%   -    0s
   0     0    9.00000    0   2     9.00000    9.00000  0.00%   -    0s

Explored 1 nodes (4977 simplex iterations) in 0.21 seconds (0.39 work units)
Thread count was 20 (of 20 available processors)

Solution count 2: 9 39

Optimal solution found (tolerance 1.00e-04)
Best objective 9.000000000000e+00, best bound 9.000000000000e+00, gap 0.00000%
```

```
optimal value = 9.0

Variable      X
-----
y(0)          1
x(32,0)        1
x(35,0)        1
x(36,0)        1
x(37,0)        1
x(40,0)        1
y(1)          1
x(0,1)         1
x(8,1)         1
x(19,1)        1
x(31,1)        1
x(49,1)        1
y(2)          1
x(1,2)         1
x(4,2)         1
x(7,2)         1
x(12,2)        1
x(14,2)        1
x(16,2)        1
x(20,2)        1
x(21,2)        1
x(24,2)        1
x(28,2)        1
x(29,2)        1
x(33,2)        1
x(39,2)        1
x(41,2)        1
y(3)          1
x(6,3)         1
x(13,3)        1
x(15,3)        1
x(23,3)        1
x(43,3)        1
x(48,3)        1
y(4)          1
x(11,4)        1
x(26,4)        1
x(27,4)        1
x(42,4)        1
x(47,4)        1
y(5)          1
x(3,5)         1
x(9,5)         1
x(17,5)        1
x(18,5)        1
y(6)          1
x(10,6)        1
x(22,6)        1
x(25,6)        1
x(34,6)        1
y(7)          1
x(2,7)         1
x(30,7)        1
x(38,7)        1
x(44,7)        1
y(8)          1
x(5,8)         1
x(45,8)        1
x(46,8)        1

gurobi>
```