

問題解決のための最適化

組合せ最適化と整数計画法

3. 1次元資材切り出し問題

堀田 敬介

# 1次元資材切り出し問題の最適化



## ➤ 1次元資材切り出し問題 1dim. cutting stock problem

- 長さ  $T$  の資材が  $m$  個ある ※資材集合  $J = \{1, \dots, m\}$
- 資材から長さが異なる複数の品を切り出したい ※品集合  $I = \{1, \dots, n\}$
- 余り部分がなるべく出ないように資材をカットする. 必要な資材はいくつか?
- 暗黙の仮定として, 切り出す品の長さは資材より短い, とする
- 1つの資材から1つの品を切り出すのが自明解 (必要な資材数 = 品数)

## ➤ 1次元資材切り出し問題のモデル化例 (ex1)

※注文品を切り出す際は, 常に鉄板と同じ方向にする (幅5mの方を揃える)

- 工場では, 幅5m × 長さ50m の鉄板を製造している
- 幅5mで長さが異なる注文(下表)を受けた

長さ(m)	3	4	5	6	7	12	15	計
個数	5	7	6	2	5	9	5	39

( $n=39$ )

( $m \leq n$ )

- 余りがなるべく出ないように鉄板をカットする. 必要(最小)鉄板数を求めよ
- 注文品の集合  $I = \{1, \dots, n\} = \{1, 2, 3, 4, 5, 6, 7, \dots, 12, 13, \dots, 18, 19, \dots, 39\}$
- 鉄板の集合  $J = \{1, \dots, m\}$       長さ3mの品(5個)   長さ4m品(7個)   長さ5m品(6個) ...
- 注文品  $i$  の長さを  $s_i$  とする ( $\mathbf{s} = (s_1, \dots, s_{39}) = (3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 5, 5, \dots, 15, 15)$ )

# 1次元資材切り出し問題の最適化

## ➤ 最適化問題の定式化(変数設定・係数表記)

➤ 0-1変数  $x_{ij} = \begin{cases} 1 & \dots \text{品}i \text{を資材}j \text{からカットする} \\ 0 & \dots \text{品}i \text{を資材}j \text{からカットしない} \end{cases}$

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

➤ 品集合  $I = \{1, \dots, n\}$

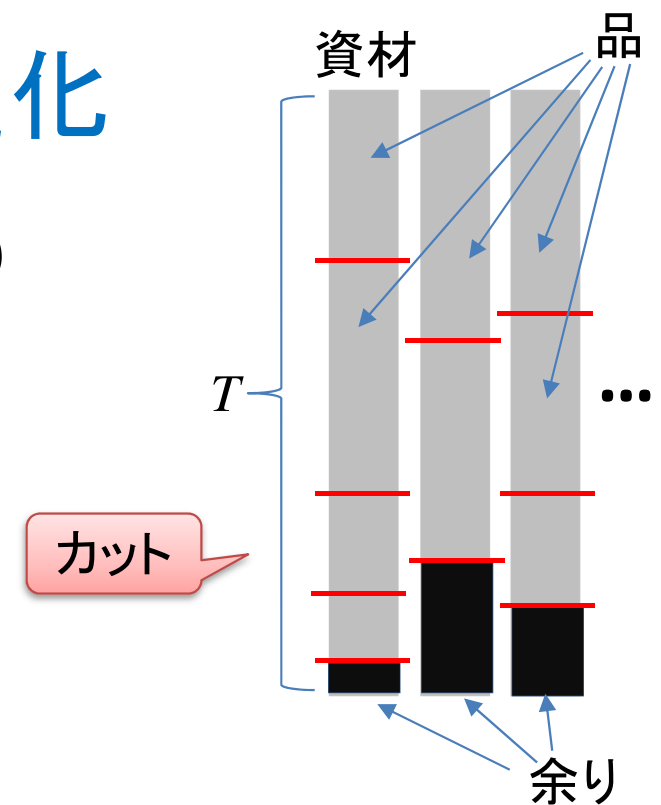
➤ 資材集合  $J = \{1, \dots, m\}$      $\ast m \leq n$

資材数は品数以下

➤ 各品  $i$  の長さを  $s_i$  とする ( $\mathbf{s} = (s_1, \dots, s_n)$ )

➤ (1つの) 資材の長さを  $T$  とする     $\ast \text{for all } i \ s_i \leq T \text{ or } \max\{s_i\} \leq T$

各品  $i$  の長さ  $s_i$  は資材長  $T$  以下  
(でないとは切り出せない)



# 1次元資材切り出し問題の最適化



## ▶ 最適化問題の定式化(ベタ表記 $\Leftrightarrow$ $\Sigma$ 表記)

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

ビンパッキング問題の  
定式化と全て同じ

# 1次元資材切り出し問題をCPLEXで解く

## ➤ 既存プロジェクトを開く

- ✓ ビンパッキング問題用に作成したプロジェクト(例: [BinPacking])を開く

## ➤ (簡易版) 詰め込み問題用のデータファイルを作成

- ✓ プロジェクト名 [BinPacking] 上で右クリック → [新規作成] → [データ] を選択
- ✓ ファイル名 [csex1.dat] として [終了] ボタンをクリック
- ✓ データファイル [csex1.dat] の中身を作成し保存

### データファイル[csex1.dat]

```
i_max = 39; // 注文数 n=39  
j_max = 10; // 鉄板数 m=10  
binB = 50; // 長さ 50m
```

```
c = [3 3 3 3 3 4 4 4 4 4 4 4 5 5 5 5 5 5 6 6 7 7 7 7 7 12 12 12 12 12 12 12 12 12 12 15 15 15 15 15];
```

長さ(m)	3	4	5	6	7	12	15	計
個数	5	7	6	2	5	9	5	39

ビンパッキング問題の定式化と全て同じなので、  
モデルファイルは [bp.mod] を流用し、  
データファイル [csex1.dat] のみを作成する

## ➤ 解く

- ✓ 実行構成 [config1] に [bp.mod] と [csex1.dat] を設定して解く

# 1次元資材切り出し問題をCPLEXで解く

## ➤ 結果([解]タブ)

必要な鉄板数は7

最適値 = 7

```
// solution (optimal) with objective 7 ←
// 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)
//
```

7.0000000000e+00

4.60000e+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 1 1 1 0 0 0];

鉄板1~7使う

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

鉄板1(計48m)

3m × 5=15m

4m × 7=28m

5m × 1=5m

鉄板2(計39m)

5m × 4=20m

6m × 2=12m

7m × 1=7m

[1 0 0 0 0 0 0 0 0 0]	3m × 5
[1 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 0]	4m × 7
[1 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 0]	
[1 0 0 0 0 0 0 0 0 0]	5m × 6
[0 1 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]	
[0 1 0 0 0 0 0 0 0 0]	
[0 1 0 0 0 0 0 0 0 0]	

変数yの値		1	1	1	1	1	1	1	0	0	0	chk
鉄板	50 m	1	2	3	4	5	6	7	8	9	10	計
注文品	3 m	5										5
	4 m	7										7
	5 m	1	5									6
	6 m		2									2
	7 m		1	4								5
	12 m			1	4	4	3					12
	15 m							2				2
	計	m	48	44	40	48	48	36	30	0	0	0
余り	m	2	6	10	2	2	14	20	50	50	50	

# 1次元資材切り出し問題を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 [csex1.dat]  
→ 生成する lpファイル [csex1.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 csex1.lp bp.mod csex1.dat [Enter]`
- この結果, モデルファイル [bp.mod] とデータファイル [csex1.dat] と同じフォルダ内に, lpファイル [csex1.lp] が出来る (※確認すること)

# 1次元資材切り出し問題を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(csex1.lp)を読み込み，model へセット

```
model = read("csex1.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("csex1.sol") [Enter]
```



# 1次元資材切り出し問題を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内で行う

(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

# 1次元資材切り出し問題をgurobiで解く(2)

➤ 問題(ex1)をpython &

```
# coding: Shift_JIS
```

```
from gurobipy import *
```

①

```
# ##### 例題設定 #####
```

```
def make_data_ex1():
```

```
    lenT = 50
```

```
    c = [3,3,3,3,3,4,4,4,4,4,4,4,5,5,5,5,5,5,6,6,7,7,7,7,7,12,12,12,12,12,12,12,12,12,12,15,15,15,15,15]
```

```
    return lenT,c
```

1つのファイル「cs.py」に

①②③の順に記述して保存

```
# ##### 実行 #####
```

```
if __name__ == "__main__":
```

```
    lenT,c = make_data_ex1()
```

```
    mod = cs(lenT,c)
```

```
    mod.write("csex1.lp")
```

```
    mod.optimize()
```

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

```
    mod.printAttr('X')
```

```
    mod.write("csex1.sol")
```

③

```
# デー
```

```
# モデ
```

```
# lpフ
```

```
# 最適
```

```
# 最適
```

```
# 最適
```

```
# ##### 定式化 #####
```

```
def cs(lenT,c):
```

```
    mod = Model("1 dimensional cutting stock problem")
```

```
# 変数設定
```

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

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

```
    for i in range(len(c)):
```

```
        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(c)):
```

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

```
for j in range(len(c)):
```

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

```
for j in range(len(c)):
```

```
    for i in range(len(c)):
```

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

```
for j in range(len(c)-1):
```

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

```
# 目的関数の設定
```

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

```
mod.update()
```

```
mod.__data = x,y
```

```
return mod
```

②

# 1次元資材切り出し問題をgurobiで解く(2)

- Pythonファイル(cs.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("cs.py").read() ) [Enter]
```

※python3系の場合

※python2系の場合の命令文は以下

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

# 1次元資材切り出し問題をgurobiで解く

## ▶ 実行結果

```
gurobi> exec(open("cs.py").read())
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<string>", line 42, in <module>
  File "<string>", line 24, in cs
TypeError: 'list' object cannot be interpreted as an integer
gurobi> exec(open("cs.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 3119 rows, 1560 columns and 65479 nonzeros
Model fingerprint: 0x1b11d6e8
Variable types: 0 continuous, 1560 integer (1560 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]
Presolve removed 1482 rows and 0 columns
Presolve time: 0.03s
Presolved: 1637 rows, 1560 columns, 6199 nonzeros
Variable types: 0 continuous, 1560 integer (1560 binary)
Found heuristic solution: objective 36.00000000

Root relaxation: objective 6.060000e+00, 2581 iterations, 0.06 seconds (0.12 work units)

   Nodes      |   Current Node   |   Objective Bounds   |   Work
  Expl Unexpl |  Obj  Depth IntInf | Incumbent  BestBd  Gap   | It/Node Time
*    0     0   |         0         | 7.00000000  7.000000  0.00% | -     0s

Explored 1 nodes (5081 simplex iterations) in 0.13 seconds (0.24 work units)
Thread count was 20 (of 20 available processors)

Solution count 2: 7 36

Optimal solution found (tolerance 1.00e-04)
Best objective 7.000000000000e+00, best bound 7.000000000000e+00, gap 0.0000%
```

optimal value = 7.0

Variable	X
y(0)	1
x(3,0)	1
x(14,0)	1
x(15,0)	1
x(21,0)	1
x(31,0)	1
x(35,0)	1
y(1)	1
x(1,1)	1
x(11,1)	1
x(22,1)	1
x(27,1)	1
x(29,1)	1
x(32,1)	1
y(2)	1
x(7,2)	1
x(10,2)	1
x(25,2)	1
x(37,2)	1
x(38,2)	1
y(3)	1
x(0,3)	1
x(4,3)	1
x(13,3)	1
x(17,3)	1
x(20,3)	1
x(28,3)	1
x(34,3)	1
y(4)	1
x(2,4)	1
x(19,4)	1
x(30,4)	1
x(33,4)	1
y(5)	1
x(6,5)	1
x(9,5)	1
x(26,5)	1
x(36,5)	1
y(6)	1
x(5,6)	1
x(8,6)	1
x(12,6)	1
x(16,6)	1
x(18,6)	1
x(23,6)	1
x(24,6)	1

gurobi>

# 【演習】1次元資材切り出し問題を解く

- ▶ 1次元資材切り出し問題の最適化(ex2)
  - ▶ 幅5m×長さ50mの鉄板を製造している
  - ▶ 幅5mで長さが異なる注文品(下表)を受け, 鉄板を切って出荷する
  - ▶ 余りになるべく出ないように鉄板をカットしたい. 鉄板はいくつ必要か?

長さ(m)	3	4	6	8	10	12	15	計
個数	7	3	4	9	3	5	7	38

# 【演習】1次元資材切り出し問題を解く

## ➤ 結果 (cplex[解]タブ)

必要な鉄板数は7

最適値 = 7

```
// solution (optimal) with objective 7 ←
// Quality Incumbent solution:
// MILP objective                               7.0000000000e+00
// MILP solution norm |x| (Total, Max)         4.50000e+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
//
```

y = [1 1 1 1 1 1 1 0 0 0]; ← 鉄板7つ使う  
x = [[1 0 0 0 0 0 0 0 0 0]

最適解

```
[1 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 0] 3m
[1 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 0]
[1 0 0 0 0 0 0 0 0 0]
[0 0 0 0 0 1 0 0 0 0] 4m
[0 0 0 0 0 1 0 0 0 0]
[1 0 0 0 0 0 0 0 0 0] 6m
[0 0 0 0 0 1 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 1 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 1 0 0 0 0 0 0] 8m
```

変数yの値		1	1	1	1	1	1	1	0	0	0	chk
鉄板	50m	1	2	3	4	5	6	7	8	9	10	計
変数xの集計	注文品	3m	7									7
		4m	1				2					3
		6m	2				2					4
		8m				1	5	2	1			9
		10m			1	2						3
		12m	1		2			1	1			5
		15m		3	1	1			2			7
	計 m	49	45	49	43	40	48	50	0	0	0	38
	余り m	1	5	1	7	10	2	0	50	50	50	

長さ(m)	3	4	6	8	10	12	15	計
個数	7	3	4	9	3	5	7	38