問題解決のための最適化

組合せ最適化と整数計画法
4. 施設配置問題

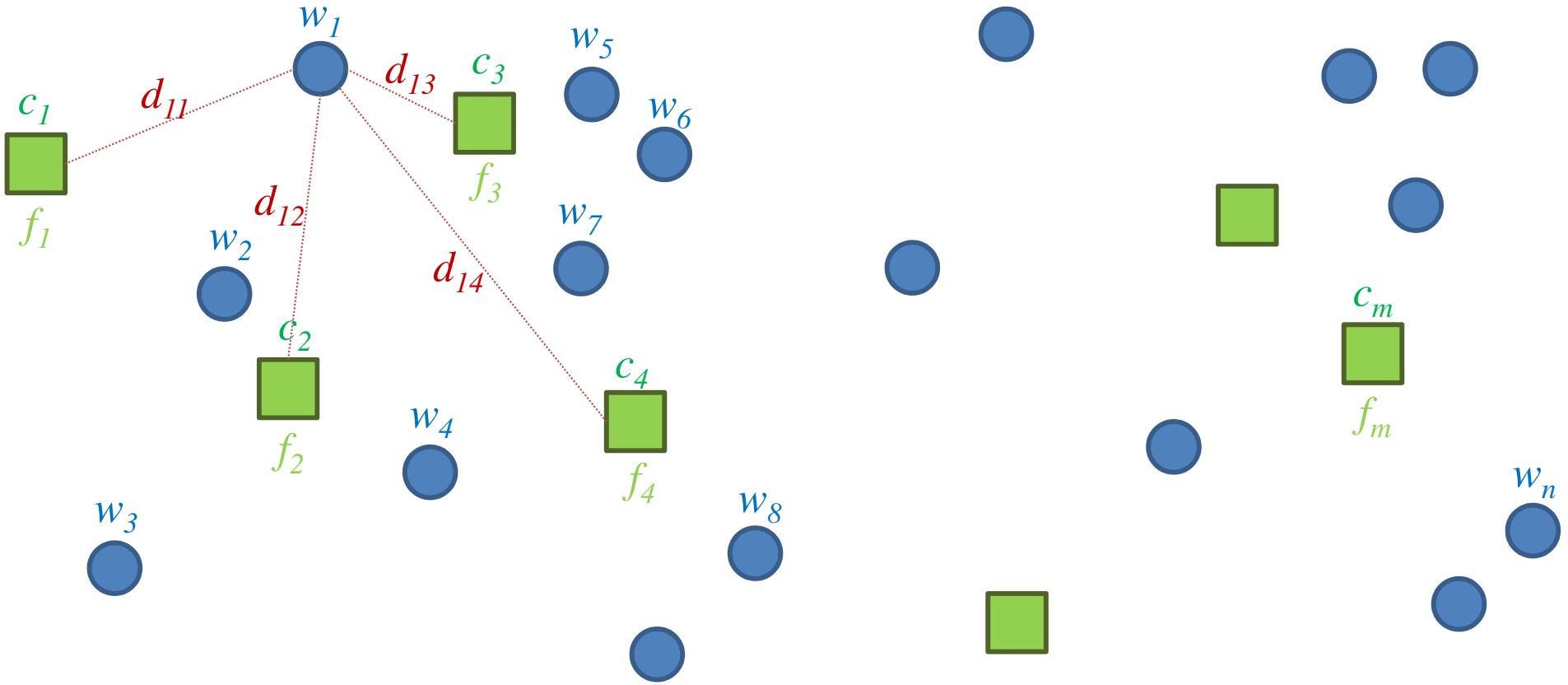
堀田 敬介

施設配置問題の最適化

＝需要点（n個）

＝施設配置候補点（m個）

- 施設配置問題 facility location problem
  - n個の需要点とm個の施設配置候補点がある
  - 需要点 i には重みw_i が，施設 j には容量 c_j と設置コスト f_j がある（所与）
  - 需要点 i が施設 j を利用する場合の利用コスト d_{ij} が所与
  - 候補点から k個選んで施設を作りたい．k は所与



施設配置問題の最適化

- 最適化問題の定式化（変数設定・係数表記）
  - 0-1変数 $x_{ij} = \begin{cases} 1 & \cdots \text{需要点} i \text{は施設} j \text{を利用} \\ 0 & \cdots \text{需要点} i \text{は施設} j \text{を利用せず} \end{cases}$
  - 0-1変数 $y_j = \begin{cases} 1 & \cdots \text{施設} j \text{を設置する} \\ 0 & \cdots \text{施設} j \text{を設置しない} \end{cases}$
  - 需要点集合 $I = \{1,\ldots,n\}$，需要点i の重み w_i（$\boldsymbol{w} = (w_1, \ldots, w_n)$）
  - 施設候補集合 $J = \{1,\ldots,m\}$，施設候補点j の容量 c_j（$\boldsymbol{c} = (c_1, \ldots, c_m)$）
  - 施設 j の設置にかかるコスト f_j（$\boldsymbol{f} = (f_1, \ldots, f_m)$）
  - 施設利用コスト行列 $\boldsymbol{D} = [d_{ij}]$
  - 設置する施設の数 k（所与）

需要点の重みを用いず変数x_{ij}を用量とする（1需要点が複数施設の利用可）定式化もあるが，ここでは1需要点は1施設のみ利用とする

施設の容量 c_j を考慮しない定式化もあるが，現実の事例では，考慮しないことの方が少ないと思うので，ここでは含めたもののみ扱う

- 施設配置問題のバリエーション（一部）
  - capacitated FLP … f_jあり，kなし，利用コスト+設置コスト最小化
  - k-median … f_jなし，kあり，利用コスト加重和の最小化（min. w.sum.）
  - k-center … f_jなし，kあり，利用コスト最大値の最小化（min. max.）

施設配置問題の最適化

- capacitated facility location(CFL)の定式化（f_jあり，kなし）

min.

$$\sum_{i=1}^{n}\sum_{j=1}^{m} d_{ij}x_{ij} + \sum_{j=1}^{m} f_j y_j$$

目的：「利用コストの和」と「施設設置費用の和」の和を最小化

s. t.

$$\sum_{j=1}^{m} x_{ij} = 1 \ (\forall i \in I)$$

制約1：任意の需要点 i は，j=1,…,m ある施設のうち丁度1施設を利用する

$$\sum_{i=1}^{n} w_i x_{ij} \leq c_j y_j \ (\forall j \in J)$$

制約2：各施設 j の容量制約．その施設 j を利用する需要点の総和は容量を c_j 以下

$$x_{ij} \leq y_j (\forall i \in I, \forall j \in J)$$

制約3：設置した施設（y_j=1）のみ利用可．（定式化の強化用制約でもある）

$$x_{ij} \in \{0,1\} \ (\forall i \in I, \forall j \in J)$$
$$y_j \in \{0,1\} \ (\forall j \in J)$$

施設配置問題の最適化

- Facility Location（k-median）の定式化（f_jなし，kあり）

min.

$$\sum_{i=1}^{n}\sum_{j=1}^{m} d_{ij}x_{ij}$$

s. t.

$$\sum_{j=1}^{m} x_{ij} = 1\ (\forall i \in I)$$

$$\sum_{i=1}^{n} w_i x_{ij} \leq c_j y_j\ (\forall j \in J)$$

$$x_{ij} \leq y_j (\forall i \in I, \forall j \in J)$$

$$\sum_{j=1}^{m} y_j = k$$

$$x_{ij} \in \{0,1\}\ (\forall i \in I, \forall j \in J)$$

$$y_j \in \{0,1\}\ (\forall j \in J)$$

目的：「利用コストの和」を最小化

制約1：任意の需要点 i は，j=1,…,m ある施設のうち丁度1施設を利用する

制約2：各施設 j の容量制約．その施設 j を利用する需要点の総和は容量を c_j 以下

制約3：設置した施設（y_j=1）のみ利用可．（定式化の強化用制約でもある）

制約4：設置する施設は k 個

施設配置問題の最適化

- Facility Location（k-center）の定式化（f_jなし，kあり）

min. t

s. t.

$$\sum_{j=1}^{m} x_{ij} = 1 \ (\forall i \in I)$$

$$\sum_{i=1}^{n} w_i x_{ij} \leq c_j y_j \ (\forall j \in J)$$

$$x_{ij} \leq y_j (\forall i \in I, \forall j \in J)$$

$$\sum_{j=1}^{m} y_j = k$$

$$d_{ij} x_{ij} \leq t \ (\forall i \in I, \forall j \in J)$$

$$x_{ij} \in \{0,1\} \ (\forall i \in I, \forall j \in J)$$

$$y_j \in \{0,1\} \ (\forall j \in J)$$

目的：最大利用コストを 最小化

制約1：任意の需要点 i は，j=1,…,m ある施設のうち丁度1施設を利用する

制約2：各施設 j の容量制約．その施設 j を利用する需要点の総和は容量を c_j 以下

制約3：設置した施設（y_j=1）のみ利用可．（定式化の強化用制約でもある）

制約4：設置する施設は k 個

制約5：全ての利用コストは t 以下
目的関数でこの t を最小化することで，最大値最小 minimax を求められる

施設配置問題の最適化

- 施設配置問facility location problem（例1）
  - 需要点 n=10，施設配置候補点m=5
  - 需要点 i と候補点 j 間の距離 d_{ij}が所与（右表）
  - 需要点の重みw_i（人口）
  - 施設容量c_j（収容数）
  - 施設設置コストf_j
  - 施設設置数 k=3カ所
  - CFL 定式化（容量ありFL）で解く
    - f_jあり，kなし
  - k-median 定式化（距離総和を最小化）で解く
    - f_jなし，kあり
  - k-center 定式化（距離最長を最小化）で解く
    - f_jなし，kあり

需要点重みw_i
施設容量c_j，設置コストf_j
需要点⇔候補点 距離 d_{ij}

	候補	1	2	3	4	5
	c_j	120	130	125	135	125
需要	f_j	90	50	70	80	60
1	30	7	6	6	9	9
2	25	3	5	7	8	5
3	40	3	7	8	5	4
4	35	2	7	1	7	9
5	25	8	7	6	7	9
6	20	3	4	2	8	4
7	30	9	9	6	5	7
8	45	2	9	7	1	5
9	30	8	9	7	7	4
10	50	8	6	6	2	1

施設配置問題をCPLEXで解く

CFL定式化
（f_jあり，kなし）

- 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：FacilityLocation）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 [FacilityLocation.mod] → [cfl.mod]　※CFLP定式化用
  - ✓ データファイル [FacilityLocation.dat] → [flex1.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

施設配置問題をCPLEXで解く

- cfl.mod

CFL定式化
（f_jあり，kなし）

```
int i_max = ...;// 需要点の数
int j_max = ...;// 施設候補点の数
int k = ...;// 配置したい施設数

range I = 1..i_max;// 需要点集合の範囲
range J = 1..j_max;// 施設候補地集合の範囲

float d[I,J] = ...;// 需要点i が施設j を使う場合の利用コスト
int w[I] = ...;// 需要点i の重み
int c[J] = ...;// 施設j の容量
int f[J] = ...;// 施設j の設置コスト

dvar int x[I,J] in 0..1;// 需要点i が施設j を利用するかどうかの0-1変数
dvar int y[J] in 0..1;// 候補地j に施設を配置するかどうかの0-1変数

minimize
  sum(i in I) sum(j in J) d[i,j]*x[i,j] + sum(j in J) f[j]*y[j];
subject to {
  forall(i in I) {
    sum(j in J) x[i,j] == 1;
  }
  forall(j in J) {
    sum(i in I) w[i]*x[i,j] <= c[j]*y[j];
  }
  forall(i in I) {
    forall(j in J) {
      x[i,j] <= y[j];
    }
  }
}
```

施設配置問題をCPLEXで解く

- flex1.dat

```
i_max = 10;
j_max = 5;
k = 3;

w = [ 30 25 40 35 25 20 30 45 30 50 ];
c = [ 95 86 99 95 85 ];
f = [ 9 5 7 8 6 ];

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

データファイル（*.dat）は，3種類の定式化で共通で使いたいので，モデルによっては使わない定数も全て記述してあることに注意

施設配置問題をCPLEXで解く

CFL定式化
（f_jあり，kなし）

- 結果（[解]タブ）

最適値 = 223

```
// solution (optimal) with objective 223
// Quality Incumbent solution:
// MILP objective                                 2.2300000000e+02
// MILP solution norm |x| (Total, Max)            1.30000e+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
// MILP x integrality error (Total, Max)          0.00000e+00
// MILP slack bound error (Total, Max)            0.00000e+00
//

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

	候補	1	2	3	4	5
	c_j	120	130	125	135	125
需要	f_j	90	50	70	80	60
1	30	7	6	6	9	9
2	25	3	5	7	8	5
3	40	3	7	8	5	4
4	35	2	7	1	7	9
5	25	8	7	6	7	9
6	20	3	4	2	8	4
7	30	9	9	6	5	7
8	45	2	9	7	1	5
9	30	8	9	7	7	4
10	50	8	6	6	2	1

施設配置問題をCPLEXで解く

k-median定式化
（f_jなし，*k*あり）

- 2つ目のモデルファイルをコピーで作成（*k*-median）
  - ✓ モデルファイル [cfl.mod]をコピー（[cfl.mod]を選択し [ctrl]+[c]キー押す）
  - ✓ ペースト（そのまま[ctrl]+[v]キー押す）
  - ✓ ファイル名を [fl-km.mod] として保存
  - ✓ 中身を修正する
- 解く
  - ✓ データファイルは共通：[flex1.dat]を使う

k-center定式化
（f_jなし，*k*あり）

- 3つ目のモデルファイルをコピーで作成（*k*-center）
  - ✓ モデルファイル [fl-km.mod]をコピー（[fl-km.mod]を選択し [ctrl]+[c]押す）
  - ✓ ペースト（そのまま [ctrl]+[v] キー押す）
  - ✓ ファイル名を [fl-cm.mod] として保存
  - ✓ 中身を修正する
- 解く
  - ✓ データファイルは共通：[flex1.dat]を使う

施設配置問題をCPLEXで解く

- fl-km.mod

```
int i_max = ...;// 需要点の数
int j_max = ...;// 施設候補点の数
int k = ...;// 配置したい施設数

range I = 1..i_max;// 需要点集合の範囲
range J = 1..j_max;// 施設候補地集合の範囲

float d[I,J] = ...;// 需要点i が施設j を使う場合の利用コスト
int w[I] = ...;// 需要点i の重み
int c[J] = ...;// 施設j の容量
int f[J] = ...;// 施設j の設置コスト

dvar int x[I,J] in 0..1;// 需要点i が施設j を利用するかどうかの0-1変数
dvar int y[J] in 0..1;// 候補地j に施設を配置するかどうかの0-1変数

minimize
  sum(i in I) sum(j in J) d[i,j]*x[i,j];
subject to {
  forall(i in I) {
    sum(j in J) x[i,j] == 1;
  }
  forall(j in J) {
    sum(i in I) w[i]*x[i,j] <= c[j]*y[j];
  }
  forall(i in I) {
    forall(j in J) {
      x[i,j] <= y[j];
    }
  }
  sum(j in J) y[j] == k;
}
```

k-median定式化
（f_jなし，*k*あり）

CFLと異なる記述

施設配置問題をCPLEXで解く

k-median定式化
（f_jなし，*k*あり）

- 結果（[解]タブ）

最適値 = 35

```
// solution (optimal) with objective 35
// Quality Incumbent solution:
// MILP objective                                3.5000000000e+01
// MILP solution norm |x| (Total, Max)           1.30000e+01  1.00000e+00
// MILP solution error (Ax=b) (Total, Max)       0.00000e+00
// MILP x bound error (Total, Max)               0.00000e+00
// MILP x integrality error (Total, Max)         0.00000e+00
// MILP slack bound error (Total, Max)           0.00000e+00
//

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

候補		1	2	3	4	5
	c_j	120	130	125	135	125
需要	f_j	90	50	70	80	60
1	30	7	6	6	9	9
2	25	3	5	7	8	5
3	40	3	7	8	5	4
4	35	2	7	1	7	9
5	25	8	7	6	7	9
6	20	3	4	2	8	4
7	30	9	9	6	5	7
8	45	2	9	7	1	5
9	30	8	9	7	7	4
10	50	8	6	6	2	1

施設配置問題をCPLEXで解く

- fl-kc.mod

```
int i_max = ...;// 需要点の数
int j_max = ...;// 施設候補点の数
int k = ...;// 配置したい施設数
range I = 1..i_max;// 需要点集合の範囲
range J = 1..j_max;// 施設候補地集合の範囲
float d[I,J] = ...;// 需要点i が施設j を使う場合の利用コスト
int w[I] = ...;// 需要点i の重み
int c[J] = ...;// 施設j の容量
int f[J] = ...;// 施設j の設置コスト

dvar int x[I,J] in 0..1;// 需要点i が施設j を利用するかどうかの0-1変数
dvar int y[J] in 0..1;// 候補地j に施設を配置するかどうかの0-1変数
dvar float t;// minimax 用変数

minimize
  t;
subject to {
  forall(i in I) {
    sum(j in J) x[i,j] == 1;
  }
  forall(j in J) {
    sum(i in I) w[i]*x[i,j] <= c[j]*y[j];
  }
  forall(i in I) {
    forall(j in J) {
      x[i,j] <= y[j];
      d[i,j]*x[i,j] <= t;
    }
  }
  sum(j in J) y[j] == k;
}
```

k-center定式化
（f_jなし，kあり）

k-medianと異なる記述

施設配置問題をCPLEXで解く

k-center定式化
（f_jなし，*k*あり）

- 結果（[解]タブ）

最適値 = 35

```
// solution (optimal) with objective 6
// Quality Incumbent solution:
// MILP objective                                6.0000000000e+00
// MILP solution norm |x| (Total, Max)           1.90000e+01  6.00000e+00
// MILP solution error (Ax=b) (Total, Max)       0.00000e+00
// MILP x bound error (Total, Max)               0.00000e+00
// MILP x integrality error (Total, Max)         0.00000e+00
// MILP slack bound error (Total, Max)           0.00000e+00
//

t = 6;
x = [[0 0 1 0 0]
     [1 0 0 0 0]
     [1 0 0 0 0]
     [1 0 0 0 0]
     [0 0 1 0 0]
     [1 0 0 0 0]
     [0 0 1 0 0]
     [0 0 0 0 1]
     [0 0 0 0 1]
     [0 0 0 0 1]];
y =  [1 0 1 0 1];
```

	候補	1	2	3	4	5
需要	c_j	120	130	125	135	125
	f_j	90	50	70	80	60
1	30	7	6	6	9	9
2	25	3	5	7	8	5
3	40	3	7	8	5	4
4	35	2	7	1	7	9
5	25	8	7	6	7	9
6	20	3	4	2	8	4
7	30	9	9	6	5	7
8	45	2	9	7	1	5
9	30	8	9	7	7	4
10	50	8	6	6	2	1

施設配置問題をgurobiで解く

- 問題（ex1）を python & gurobi で記述（データ生成部分①）

①

```
# coding: Shift_JIS
from gurobipy import *

# ######## 例題設定 #######
def make_data_ex1():
    k = 3                                   # 必要施設数(設置施設数)
    w = [30,25,40,35,25,20,30,45,30,50]     # 需要点iの重み
    c = [120,130,125,135,125]               # 施設jの容量
    f = [90,50,70,80,60]                    # 施設jの設置コスト
    d = {(0,0):7,(0,1):6,(0,2):6,(0,3):9,(0,4):9,
            (1,0):3,(1,1):5,(1,2):7,(1,3):8,(1,4):5,
            (2,0):3,(2,1):7,(2,2):8,(2,3):5,(2,4):4,
            (3,0):2,(3,1):7,(3,2):1,(3,3):7,(3,4):9,
            (4,0):8,(4,1):7,(4,2):6,(4,3):7,(4,4):9,
            (5,0):3,(5,1):4,(5,2):2,(5,3):8,(5,4):4,
            (6,0):9,(6,1):9,(6,2):6,(6,3):5,(6,4):7,
            (7,0):2,(7,1):9,(7,2):7,(7,3):1,(7,4):5,
            (8,0):8,(8,1):9,(8,2):7,(8,3):7,(8,4):4,
            (9,0):8,(9,1):6,(9,2):6,(9,3):2,(9,4):1,
    } # 需要点iが施設jを使う場合の利用コスト
    return k,w,c,f,d
```

※3種の定式化で共通で使うので，それぞれで使うデータと使わないデータがあることに注意

施設配置問題をgurobiで解く

CFL定式化
（f_jあり，kなし）

- 問題（ex1）を python &

1つのファイル「cfl.py」に
①②③の順に記述して保存

②

```
# ######## 定式化 ########
def cfl(w,c,f,d):
    mod = Model("facility location problem:CFL")

    # 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(w)):
            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(w)):
        mod.addConstr(quicksum(x[i,j] for j in range(len(c))) == 1)
    for j in range(len(c)):
        mod.addConstr(quicksum(w[i]*x[i,j] for i in range(len(w))) <= c[j]*y[j])
    for j in range(len(w)):
        for i in range(len(c)):
            mod.addConstr(x[i,j] <= y[j])

    # 目的関数の設定
    mod.setObjective(quicksum(d[i,j]*x[i,j] for i in range(len(w)) for j in
range(len(c))) + quicksum(f[j]*y[j] for j in range(len(c))), GRB.MINIMIZE)
    mod.update()
    mod.__data = x,y
    return mod
```

③

```
# ######## 実行 ########
if __name__=="__main__":
    k,w,c,f,d = make_data_ex1()        # デー
    mod = cfl(w,c,f,d)                 # モテ
    mod.write("cflex1.lp")             # lpフ
    mod.optimize()                     # 最適
    print("¥n optimal value = ", mod.ObjVa
    mod.printAttr('X')                 # 最適
    mod.write("cflex1.sol")            # 最適解をsolファイルに出力
```

施設配置問題をgurobiで解く

- Pythonファイル（cfl.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(“cfl.py”).read() )  [Enter]
```

※python3系の場合

※python2系の場合の命令文は以下

```
gurobi> execfile(“cfl.py”)  [Enter]
```

施設配置問題をgurobiで解く

CFL定式化
（f_jあり，kなし）

- 実行結果

```
gurobi> exec(open("cfl.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 65 rows, 55 columns and 205 nonzeros
Model fingerprint: 0x4333b926
Variable types: 0 continuous, 55 integer (55 binary)
Coefficient statistics:
  Matrix range     [1e+00, 1e+02]
  Objective range  [1e+00, 9e+01]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 1e+00]
Found heuristic solution: objective 404.0000000
Presolve time: 0.00s
Presolved: 65 rows, 55 columns, 205 nonzeros
Variable types: 0 continuous, 55 integer (55 binary)
Found heuristic solution: objective 385.0000000

Root relaxation: objective 1.995250e+02, 41 iterations, 0.00 seconds (0.00 work

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0  199.52500    0   13  385.00000  199.52500  48.2%     -    0s
H    0     0                       223.0000000  199.52500  10.5%     -    0s
     0     0     cutoff    0       223.00000  223.00000  0.00%     -    0s

Cutting planes:
  Gomory: 5
  MIR: 1
  StrongCG: 1
  Zero half: 4

Explored 1 nodes (48 simplex iterations) in 0.01 seconds (0.00 work units)
Thread count was 20 (of 20 available processors)

Solution count 3: 223 385 404

Optimal solution found (tolerance 1.00e-04)
Best objective 2.230000000000e+02, best bound 2.230000000000e+02, gap 0.0000%
```

```
optimal value =  223.0

   Variable            X
-----------------------
       y(1)            1
     x(0,1)            1
     x(1,1)            1
     x(2,1)            1
       y(2)            1
     x(3,2)            1
     x(4,2)            1
     x(5,2)            1
     x(6,2)            1
       y(4)            1
     x(7,4)            1
     x(8,4)            1
     x(9,4)            1
gurobi>
```

施設配置問題をgurobiで解く

k-median定式化
（*f*なし，*k*あり）

- 問題（ex1）を python &

1つのファイル「fl-km.py」に
①②③の順に記述して保存

```
# ######## 定式化 ########
def flkm(k,w,c,d):
    mod = Model("facility location problem: FL_k-median")

    # 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(w)):
            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(w)):
        mod.addConstr(quicksum(x[i,j] for j in range(len(c))) == 1)
    for j in range(len(c)):
        mod.addConstr(quicksum(w[i]*x[i,j] for i in range(len(w))) <= c[j]*y[j])
    for j in range(len(w)):
        for i in range(len(c)):
            mod.addConstr(x[i,j] <= y[j])
    mod.addConstr(quicksum(y[j] for j in range(len(c))) == k)

    # 目的関数の設定
    mod.setObjective(quicksum(d[i,j]*x[i,j] for i in range(len(w)) for j in
range(len(c))), GRB.MINIMIZE)
    mod.update()
    mod.__data = x,y
    return mod
```

②

```
# ######## 実行 ########
if __name__=="__main__":
    k,w,c,f,d = make_data_ex1()        # デー
    mod = flkm(k,w,c,d) # モデルの生成
    mod.write("fl-kmex1.lp")           # lpフ
    mod.optimize()                     # 最適
    print("¥n optimal value = ", mod.ObjVa
    mod.printAttr('X')                 # 最適
    mod.write("fl-kmex1.sol")          # 最適
```

③

施設配置問題をgurobiで解く

k-median定式化
（f_fなし，*k*あり）

- 実行結果

```
optimal value =  35.0

   Variable            X
-----------------------
       y(0)            1
     x(1,0)            1
     x(2,0)            1
     x(7,0)            1
       y(2)            1
     x(0,2)            1
     x(3,2)            1
     x(4,2)            1
     x(5,2)            1
       y(4)            1
     x(6,4)            1
     x(8,4)            1
     x(9,4)            1
gurobi>
```

```
gurobi> exec(open("fl-km.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 66 rows, 55 columns and 210 nonzeros
Model fingerprint: 0x980ec5f3
Variable types: 0 continuous, 55 integer (55 binary)
Coefficient statistics:
  Matrix range     [1e+00, 1e+02]
  Objective range  [1e+00, 9e+00]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 3e+00]
Found heuristic solution: objective 54.0000000
Presolve time: 0.00s
Presolved: 66 rows, 55 columns, 210 nonzeros
Variable types: 0 continuous, 55 integer (55 binary)
Found heuristic solution: objective 51.0000000

Root relaxation: objective 3.445238e+01, 23 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   34.45238    0    4   51.00000   34.45238  32.4%     -    0s
H    0     0                       42.0000000   34.45238  18.0%     -    0s
H    0     0                       36.0000000   34.45238  4.30%     -    0s
H    0     0                       35.0000000   34.45238  1.56%     -    0s
     0     0   34.45238    0    4   35.00000   34.45238  1.56%     -    0s

Explored 1 nodes (23 simplex iterations) in 0.01 seconds (0.00 work units)
Thread count was 20 (of 20 available processors)

Solution count 5: 35 36 42 ... 54

Optimal solution found (tolerance 1.00e-04)
Best objective 3.500000000000e+01, best bound 3.500000000000e+01, gap 0.0000%
```

施設配置問題をgurobiで解く

k-center定式化
（f_jなし，kあり）

- 問題（ex1）を python &

1つのファイル「fl-kc.py」に
①②③の順に記述して保存

②

```
# ######## 定式化 ########
def flkc(k,w,c,d):
    mod = Model(“facility location problem: FL_k-center")

    # 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(w)):
            x[i,j] = mod.addVar(vtype="B", name="x(%s,%s)" % (i,j))
    t = mod.addVar(vtype="C", name="t")
    mod.update()

    # 制約条件の設定
    for i in range(len(w)):
        mod.addConstr(quicksum(x[i,j] for j in range(len(c))) == 1)
    for j in range(len(c)):
        mod.addConstr(quicksum(w[i]*x[i,j] for i in range(len(w))) <= c[j]*y[j])
    for j in range(len(w)):
        for i in range(len(c)):
            mod.addConstr(x[i,j] <= y[j])
            mod.addConstr(d[i,j]*x[i,j] <= t)
    mod.addConstr(quicksum(y[j] for j in range(len(c))) == k)

    # 目的関数の設定
    mod.setObjective(t, GRB.MINIMIZE)
    mod.update()
    mod.__data = x,y
    return mod
```

③

```
# ######## 実行 ########
if __name__=="__main__":
    k,w,c,f,d = make_data_ex1()        # デー
    mod = flkc(k,w,c,d)   # モデルの生成
    mod.write(“fl-kcex1.lp")           # lpフ
    mod.optimize()                     # 最適
    print("¥n optimal value = ", mod.ObjVa
    mod.printAttr('X')                 # 最適
    mod.write(“fl-kcex1.sol")          # 最適
```

施設配置問題をgurobiで解く

k-center定式化
（f_jなし，kあり）

- 実行結果

```
gurobi> exec(open("fl-kc.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 116 rows, 56 columns and 310 nonzeros
Model fingerprint: 0xf99b88f6
Variable types: 1 continuous, 55 integer (55 binary)
Coefficient statistics:
  Matrix range     [1e+00, 1e+02]
  Objective range  [1e+00, 1e+00]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 3e+00]
Found heuristic solution: objective 9.0000000
Presolve removed 30 rows and 0 columns
Presolve time: 0.00s
Presolved: 86 rows, 56 columns, 255 nonzeros
Variable types: 0 continuous, 56 integer (55 binary)

Root relaxation: objective 6.000000e+00, 23 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    6.00000    0    2    9.00000    6.00000  33.3%     -    0s
H    0     0                       7.0000000    6.00000  14.3%     -    0s
H    0     0                       6.0000000    6.00000  0.00%     -    0s
     0     0    6.00000    0    2    6.00000    6.00000  0.00%     -    0s

Explored 1 nodes (23 simplex iterations) in 0.02 seconds (0.00 work units)
Thread count was 20 (of 20 available processors)

Solution count 3: 6 7 9

Optimal solution found (tolerance 1.00e-04)
Best objective 6.000000000000e+00, best bound 6.000000000000e+00, gap 0.0000%
```

```
optimal value =  6.0

   Variable            X
-----------------------
       y(0)            1
     x(1,0)            1
     x(3,0)            1
     x(7,0)            1
       y(2)            1
     x(0,2)            1
     x(4,2)            1
     x(5,2)            1
     x(6,2)            1
       y(4)            1
     x(2,4)            1
     x(8,4)            1
     x(9,4)            1
         t             6
gurobi>
```

【演習】施設配置問題を解く

- 施設配置問facility location problem（ex2）
  - 需要点 n=30，施設配置候補点m=10
  - 需要点 i と候補点 j 間の距離 d_{ij}が所与（右表）
  - 需要点の重みw_i（人口）
  - 施設容量c_j（収容数）
  - 施設設置コストf_j
  - 施設設置数 k=4カ所
  - CFL 定式化（容量ありFL）で解く
    - f_jあり，kなし
  - k-median 定式化（距離総和を最小化）で解く
    - f_jなし，kあり
  - k-center 定式化（距離最長を最小化）で解く
    - f_jなし，kあり

需要点重みw_i
施設容量c_j，設置コストf_j
需要点⇔候補点 距離 d_{ij}

	候補	1	2	3	…	10
需要	c_j					
	f_j					
1						
2						
3						
4						
5						
6						
7						
8						
…						
30						