システム数理 II ：期末レポート
情報科学類　3年
201811319
永崎遼太

自分の身近にあるややこしい問題を説明し，その解決のために数理最適化など工学的手法をどのように用いることができるか，また用いた結果について報告しなさい.

線形最適化問題を用いる実際に役に立つ応用問題として、学校生活では重要な問題である
クラス編成問題(class assignment problem)を考える。
クラス編成問題とは、具体的には次のような設定の問題である。

問題設定：
大学の研究室振り分けを考える。生徒は学生IDがS001からS050までの50名である。研究室はR01からR10まであるとする。各学生の研究室配属に対する第一から第四希望までの順位を書いた次のようなリストが、データファイルとして手元にあるとする。

リスト(データファイルの一部)：

	R01	R02	R03	R04	R05	R06	R07	R08	R09	R10
S001		1		4		2		3		
S002			1	3			2			4
S003	4		2			2			1	
S004		3			4		2			1
S005						1	4	3	2	
S006	2		4	2		1				
S007	1				3		4		2	
S008	1	4		3		2				
S009							1	2	3	1
S010				2	3	1		4		

行が学生IDに対応し、列が研究室に対応する表になっており、学生 i がクラス j を第 k 希望目に希望している時、i 行 j 列の値が k となっており、それ以外空白である。

各研究室の学生数が均等になるように、各学生の希望をなるべく叶える形で学生を研究室に配属させたい。

クラス編成問題は線形最適化問題として定式化して解くことができ、様々な方法が提案されている。本レポートでは、クラス編成問題を数理最適化を用いて解き、その結果をまとめる。

定式化

変数の割り当て

学生の集合をI = {S001, S002, , ⋯ , S050}とし、研究室の集合をJ = {R01, R02, ⋯ , R10}とする。学生 $i \in I$と研究室 $j \in J$ の全ての組み合わせに対して変数 x_{ij} を用意する。この変数は、学生 $i \in I$ が研究室 $j \in J$ に割り当てられた場合に1、それ以外は0の値を取るものと考える。

目的関数

学生の満足度を最大にすることを考える。学生 i が第一希望の研究室に所属した場合その学生の満足度を100とし、第二希望の研究室に所属した場合に50、第三希望の場合20、第四希望の場合0、そして希望外の時、-100000とする。つまり、満足度関数 p_{ij} を、次のようにして定義する。

$$p_{i,j} = \begin{cases} 100 & \text{student i's first choice is lab j} \\ 50 & \text{student i's second choice is lab j} \\ 20 & \text{student i's third choice is lab j} \\ 0 & \text{Student i's fourth choice is lab j} \\ -100000 & \text{otherwise} \end{cases}$$

目的関数は、全ての学生の満足度の総和、つまり

$$\sum_{i \in I} \sum_{j \in J} p_{i,j} x_{i,j}$$

であり、これを最大化する。

制約条件

次の2つについて考える。

- どの学生も1つの研究室に配属される
- 各クラスの人数は、5人である(50/10=5)

これらを数式で表すと、次のようになる。

$$\sum_{j \in J} x_{i,j} = 1, \forall i \in I$$

$$\sum_{i \in I} x_{i,j} = 5, \forall j \in J$$

以上をまとめると、学生の満足度最大化クラス編成問題は以下のような線形最適化問題となる。

$$\text{maximize} \sum_{j \in J} p_{i,j} x_{i,j} = 1$$

$$subject\ to \begin{cases} \sum_{j \in J} x_{i,j} = 1, \forall i \in I \\ \\ \sum_{i \in I} x_{i,j} = 5, \forall j \in J \end{cases}$$

変数 $x_{i,j}$ は、学生 $i \in I$ が研究室 $j \in J$ に割り当てられる時1、そうでない時0を取るので、$x_{i,j} \in \{0,1\}$ とすべきだが、整数値になる最適解が必ず存在する音が保証されており、そのような条件を必要としない。

計算機実験 / Pythonによる実装

クラス編成問題を実データを使って解く。データはCSVファイル(data.csv)に出力され用いられる。以下では、Pythonを用いて作成したプログラムを機能ごとに分解し、処理の順番にそれらを説明していく。
実際に作成した機能は、次のようになっている。

---

1. データの読み込み

---

2. データの整合性のチェック

---

3. データの分析

---

4. クラス編成最適化

---

5. 最適化結果の分析

次ページからそれぞれについて説明する。

1. データの読み込み

次のプログラムを作成した。

```
%matplotlib inline
import pandas as pd

df=pd.read_csv('data.csv',index_col=0)
df
```

2行目でpandasパッケージをpdという省略形で読み込み、4行目でdata.csvをdfという変数に、データフレームで読み込み代入している。5行目で、dfを表示している。出力結果の一部は次のようになる。

	R01	R02	R03	R04	R05	R06	R07	R08	R09	R10
S001	NaN	1.0	NaN	4.0	NaN	2.0	NaN	3.0	NaN	NaN
S002	NaN	NaN	1.0	3.0	NaN	NaN	2.0	NaN	NaN	4.0
S003	4.0	NaN	2.0	NaN	NaN	3.0	NaN	NaN	1.0	NaN
S004	NaN	3.0	NaN	NaN	4.0	NaN	2.0	NaN	NaN	1.0
S005	NaN	NaN	NaN	NaN	NaN	1.0	4.0	3.0	2.0	NaN
S006	3.0	NaN	4.0	2.0	NaN	1.0	NaN	NaN	NaN	NaN
S007	1.0	NaN	NaN	NaN	3.0	NaN	4.0	NaN	2.0	NaN
S008	1.0	4.0	NaN	3.0	NaN	2.0	NaN	NaN	NaN	NaN
S009	NaN	NaN	NaN	NaN	NaN	NaN	1.0	4.0	3.0	2.0
S010	NaN	NaN	NaN	2.0	3.0	1.0	NaN	4.0	NaN	NaN

空白の部分にはNaN(Not a Number)という値が代入されている。

2. データの整合性のチェック

次のプログラムを作成した。

```
for i in df.index:
    if (df.loc[i]==1).sum()==1 and \
        (df.loc[i]==2).sum()==1 and \
        (df.loc[i]==3).sum()==1 and \
        (df.loc[i]==4).sum()==1:
        print(i,'ok')
    else:
        print(i,'NG')
```

各学生の第一から第四希望のクラスが1つずつである場合、学生IDと 'ok' という文字列を出力を返し、そうではない場合、学生IDと 'NG' という文字列を出力する。出力結果の一部は次のようになる。

```
S001 ok
S002 ok
S003 ok
S004 ok
S005 ok
S006 ok
S007 ok
S008 ok
S009 ok
S010 ok
```

出力結果より、S001からS050までの全ての学生について、第一希望から第四希望までを1つずつ選択していることがわかり、入力データの整合性が確認された。

3. データの分析

次のプログラムを作成した。

```
d = {j: [(df.loc[:,j]==1).sum(),(df.loc[:,j]==2).sum(),
         (df.loc[:,j]==3).sum(),(df.loc[:,j]==4).sum(),
        (df.loc[:,j]> 0).sum()] for j in df.columns}
df2 = pd.DataFrame(d,
                   index=['第1希望', '第2希望','第3希望','第4希望','合計'])
df2
```

データフレームdfの各列に対して、それぞれ希望を表す数値1, 2, 3, 4が何個入っているかを数え、さらに合計を計算し、新たなデータフレームdfを作成している。これは、各研究室について、第

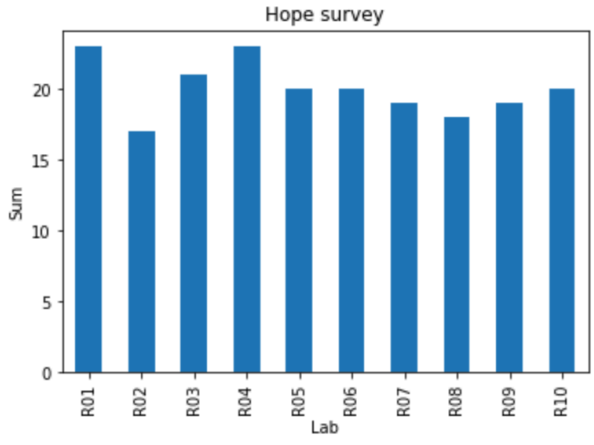
一から第四希望までの学生が何人いるかを数え、さらに合計を取るという処理を実行している。6行目で新たに作成したdf2を表示している。出力結果は次のようになる。

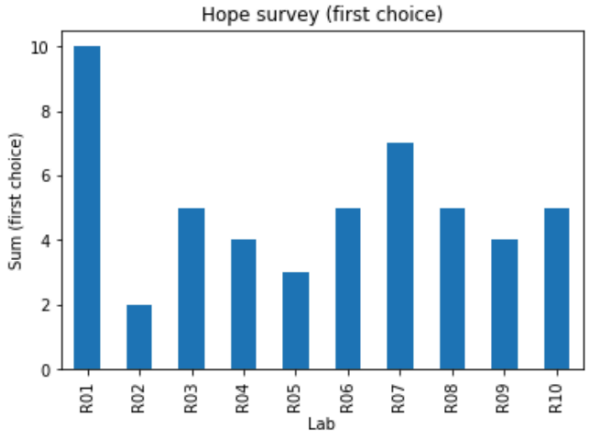
	R01	R02	R03	R04	R05	R06	R07	R08	R09	R10
第1希望	10	2	5	4	3	5	7	5	4	5
第2希望	4	2	4	7	5	5	4	4	7	8
第3希望	4	7	7	5	5	6	4	5	4	3
第4希望	5	6	5	7	7	4	4	4	4	4
合計	23	17	21	23	20	20	19	18	19	20

これより、研究室の希望の詳細を分析することができる。また、このデータを可視化するために、次のコードを追加した。

```
import matplotlib.pyplot as plt
%matplotlib inline
df2.loc['合計'].plot(kind='bar')
plt.title('Hope survey')
plt.xlabel('Lab')
plt.ylabel('Sum')
plt.show()
```

このコードは、データフレームdf2の’合計‘の項目を棒グラフで表すためのコードである。出力結果は次の左図のようになる。





さらに、上のコードの’合計’の箇所を’第一希望’と変え、各研究室への第一希望だけの合計についての棒グラフを作成すると上の右図のようになる。

出力された棒グラフにより可視化されることで、学生の希望の分布が把握でき、人気の研究室などを視覚的に確認することができた。

4. クラス編成最適化

次のプログラムを作成した。

```
from pulp import *
from itertools import product # くり返しのため
import math # floor, ceil関数のため
MEPS=1.0e-6
```

A

```
n=len(df.index) # 学生の数
m=len(df.columns) # クラスの数
lb=math.floor(n/m) # 1クラス人数の下限
ub=math.ceil(n/m) # 1クラス人数の上限
```

```
# 満足度
score={1:100,2:50,3:30,4:0}
ngscore=-100000 # 希望外
```

B

```
prob=LpProblem('ClassAssgnment',sense=LpMaximize)
x={}
p={}
for i,j in product(df.index,df.columns):
    x[i,j]=LpVariable('x('+str(i)+','+str(j)+')',lowBound=0)
    p[i,j]=score[int(df.loc[i,j])] if df.loc[i,j]>MEPS else ngscore
```

C

```
# 目的関数
prob += lpSum(p[i,j]*x[i,j] for i, j in product(df.index,df.columns))
```

```
# 制約式
for i in df.index:
    prob += lpSum(x[i,j] for j in df.columns)==1
for j in df.columns:
    prob += lpSum(x[i,j] for i in df.index) >= lb
    prob += lpSum(x[i,j] for i in df.index) <= ub
```

D

```
# 最適化
prob.solve()
# 解の出力
print(LpStatus[prob.status])
print('学生の満足度の総計は', int(value(prob.objective)))
print('学生一人当たりの平均満足度は', int(value(prob.objective))/n)
```

E

これを5つに分割したA〜Eについて、それぞれ説明をする。

A

```
from pulp import *
from itertools import product # くり返しのため
import math # floor, ceil関数のため
MEPS=1.0e-6
```

必要なパッケージの読み込みを行なっている。また、4行目ではマシンイプシロンを設定している。

B

```
n=len(df.index) # 学生の数
m=len(df.columns) # クラスの数
lb=math.floor(n/m) # 1クラス人数の下限
ub=math.ceil(n/m) # 1クラス人数の上限
```

```
# 満足度
score={1:100,2:50,3:30,4:0}
ngscore=-100000 # 希望外
```

学生の数、クラス(研究室)の数、クラス定員の上下限の設定を行なっている。そして、希望順ごとの満足度を設定している。希望順位がキーで、値が満足度となる辞書を用いて満足度を定義している。希望外の場合は-100000としている。このようにすることで、希望外の学生が一人でもいる編成の場合には全体満足度が負になるため、気付きやすくなっている。

C

```
prob=LpProblem('ClassAssgnment',sense=LpMaximize)
x={}
p={}
for i,j in product(df.index,df.columns):
    x[i,j]=LpVariable('x('+str(i)+','+str(j)+')',lowBound=0)
    p[i,j]=score[int(df.loc[i,j])] if df.loc[i,j]>MEPS else ngscore
```

一行目に最適化モデルの設定を行なっている。以降は変数 $x_{i,j}$ と満足度関数 $p_{i,j}$ の定義である。

データフレームdfの、全ての行と列の組み合わせに対して変数を割り当て、さらにその位置に数値が入っている場合その数値をキーとする満足度を満足度関数値としている。

D

```
# 目的関数
prob += lpSum(p[i,j]*x[i,j] for i, j in product(df.index,df.columns))
```

```
# 制約式
for i in df.index:
    prob += lpSum(x[i,j] for j in df.columns)==1
for j in df.columns:
    prob += lpSum(x[i,j] for i in df.index) >= lb
    prob += lpSum(x[i,j] for i in df.index) <= ub
```

目的関数と制約式を定義している。目的関数は、変数とそれに対応する満足度関数の積の総和である。制約条件は、各クラス(研究室)に上限、下限がある中、各学生はどれか1つのクラスに割り当てられるということを表している。

E

```
# 最適化
prob.solve()
# 解の出力
print(LpStatus[prob.status])
print('学生の満足度の総計は', int(value(prob.objective)))
print('学生一人当たりの平均満足度は', int(value(prob.objective))/n)
```

最適解の求解と、結果の表示を行なっている。

以上A〜Eを順に実行していくと、最終的に次の結果が得られる。

```
Optimal
学生の満足度の総計は 4550
学生一人当たりの平均満足度は 91.0
```

これは、最適解が得られたことを表し、その解について、学生の満足度の総計及び学生一人当たりの平均満足度を計算し出力している。この結果からわかるように、全ての学生が第四希望までの研究室に配属され、また学生一人当たりの平均満足度が91.0と高い値を示していることから、良いクラス編成が行われたということがわかる。

5. 最適化結果の分析

4で得られた最適解の詳細の情報を表示するために、次のプログラムを作成した。

```
# 配属の詳細とクラス別満足度
dfr=df.copy()
for i,j in product(dfr.index,dfr.columns):
    dfr.loc[i,j]=df.loc[i,j] if x[i,j].varValue > MEPS else 0

d2={j:[(dfr.loc[:,j]==1).sum(),(dfr.loc[:,j]==2).sum(),
      (dfr.loc[:,j]==3).sum(),(dfr.loc[:,j]==4).sum(),
      (dfr.loc[:,j]>0).sum(),
       sum([p[i,j] for i in df.index
            if x[i,j].varValue > MEPS])] for j in dfr.columns}
df3=pd.DataFrame(d2,
                 index=['第1希望', '第2希望','第3希望','第4希望','合計','クラス満足度'])
df3
```

このプログラムでは、各クラス(研究室)別に、第一から第四希望の学生の配属人数及びクラスの総合満足度を新たなデータフレームdf3に格納する。このプログラムの出力結果は次のようになる。

	R01	R02	R03	R04	R05	R06	R07	R08	R09	R10
第1希望	5	2	5	4	3	5	5	5	4	5
第2希望	0	1	0	1	2	0	0	0	1	0
第3希望	0	0	0	0	0	0	0	0	0	0
第4希望	0	2	0	0	0	0	0	0	0	0
合計	5	5	5	5	5	5	5	5	5	5
クラス満足度	500	250	500	450	400	500	500	500	450	500

出力結果より、全ての学生が第4希望までの研究室に配属されていることがわかる。この時、希望順位が高い研究室に配属された学生が多いことが確認できる。

以上より、自分の身近にあるややこしい問題として取り上げたクラス編成問題について、線形最適化問題として定式化を行い、数理最適化を用いて最適な結果として解を求めることができた。

次ページから、異なる入力に対して実行した結果をまとめている。

本問題の設定と同様の問題について、異なる入力データについて実行した結果は以下のようになった。

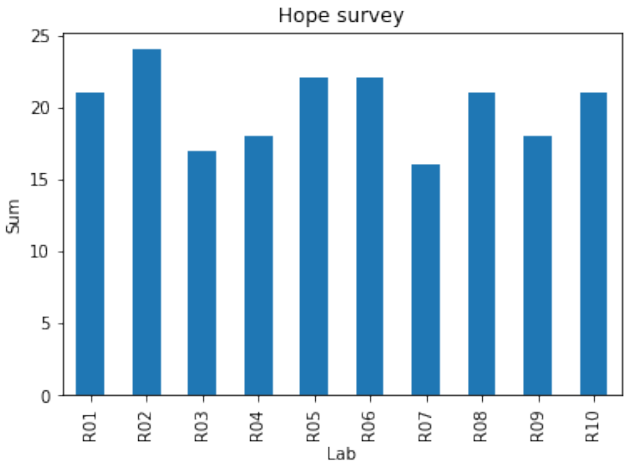
1. データの読み込み

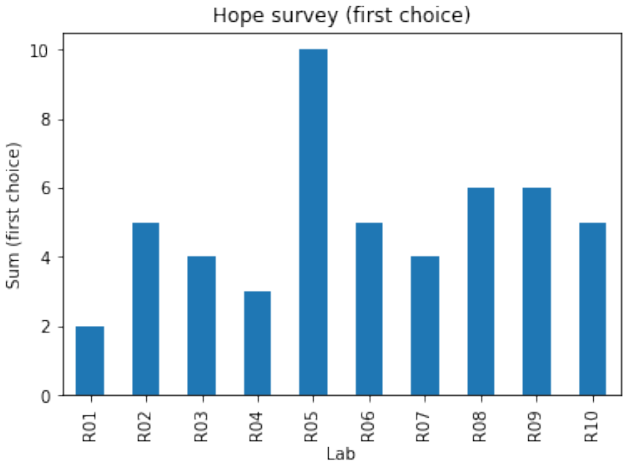
	R01	R02	R03	R04	R05	R06	R07	R08	R09	R10
第1希望	2	5	4	3	10	5	4	6	6	5
第2希望	6	8	6	4	2	6	5	6	4	3
第3希望	7	6	4	6	5	4	3	6	2	7
第4希望	6	5	3	5	5	7	4	3	6	6
合計	21	24	17	18	22	22	16	21	18	21

2. データの整合性のチェック

問題なし。

3. データの分析





4. クラス編成最適化

```
Optimal
学生の満足度の総計は 4570
学生一人当たりの平均満足度は 91.4
```

5. 最適化結果の分析

	R01	R02	R03	R04	R05	R06	R07	R08	R09	R10
第1希望	2	5	4	3	5	5	4	5	5	5
第2希望	2	0	0	1	0	0	0	0	0	0
第3希望	1	0	1	1	0	0	1	0	0	0
第4希望	0	0	0	0	0	0	0	0	0	0
合計	5	5	5	5	5	5	5	5	5	5
クラス満足度	330	500	430	380	500	500	430	500	500	500

正しく最適解を求めることができていることが確認できる。

生徒数を100人に拡張した場合の結果は次のようなった。

1. データの読み込み

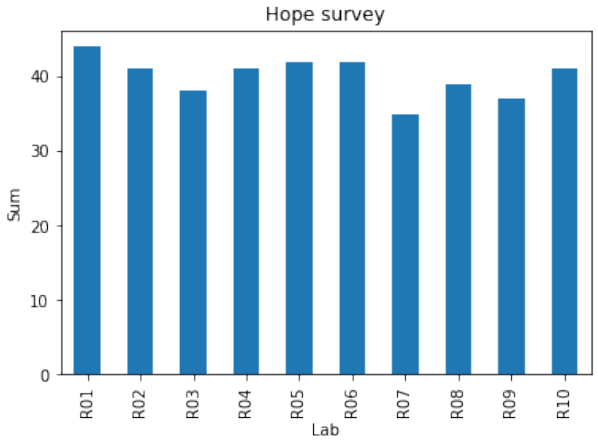
	R01	R02	R03	R04	R05	R06	R07	R08	R09	R10
S001	NaN	2.0	NaN	NaN	1.0	3.0	NaN	NaN	NaN	4.0
S002	NaN	NaN	2.0	NaN	NaN	NaN	4.0	NaN	1.0	3.0
S003	NaN	3.0	NaN	4.0	NaN	NaN	NaN	1.0	2.0	NaN
S004	4.0	NaN	2.0	NaN	3.0	NaN	1.0	NaN	NaN	NaN
S005	NaN	1.0	4.0	NaN	NaN	3.0	NaN	2.0	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...
S096	1.0	2.0	4.0	NaN	NaN	NaN	NaN	NaN	3.0	NaN
S097	NaN	NaN	NaN	2.0	3.0	NaN	1.0	NaN	NaN	4.0
S098	NaN	NaN	1.0	NaN	NaN	4.0	NaN	2.0	3.0	NaN
S099	1.0	NaN	3.0	NaN	4.0	NaN	NaN	NaN	NaN	2.0
S100	NaN	NaN	4.0	NaN	1.0	3.0	NaN	NaN	2.0	NaN

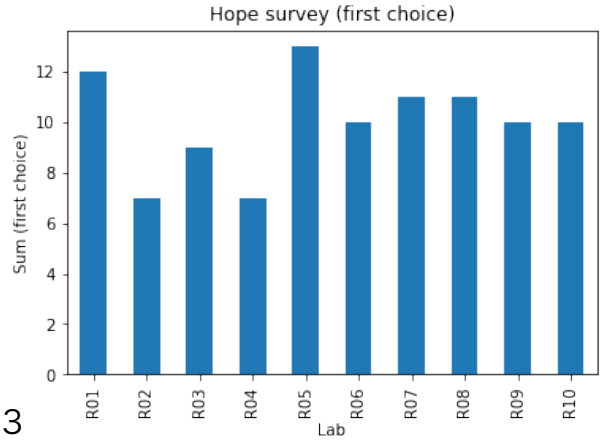
100 rows × 10 columns

2. データの整合性のチェック

問題なし。

3. データの分析





4. クラス編成最適化

```
Optimal
学生の満足度の総計は 9650
学生一人当たりの平均満足度は 96.5
```

5. 最適化結果の分析

	R01	R02	R03	R04	R05	R06	R07	R08	R09	R10
第1希望	10	7	9	7	10	10	10	10	10	10
第2希望	0	3	1	3	0	0	0	0	0	0
第3希望	0	0	0	0	0	0	0	0	0	0
第4希望	0	0	0	0	0	0	0	0	0	0
合計	10	10	10	10	10	10	10	10	10	10
クラス満足度	1000	850	950	850	1000	1000	1000	1000	1000	1000

正しく最適解を求めることができていることが確認できる。