身近な問題を解く！組合せ最適化！

東京理科大学　理学部第一部　応用数学科　講師　胡艷楠

1　はじめに

最適化問題とは，定められた指標をさまざまな条件のもとで最大（または最小）にする解を求める問題です．最適化問題の中でも，その解が集合や組合せ等のように離散的に表現される問題は組合せ最適化問題と呼ばれます．私たちは毎日たくさんの組合せ最適化問題に直面しています．

まず以下の学校の教室を予約する問題を考えましょう．いくつかの部活はある日にある教室を利用した活動を希望しています．各部活は教室を利用する開始時刻と終了時刻があります．お互いに両立できる，つまり，利用時間が重ならないようにしながら教室を利用する部活の数を最大にするにはどのようにすれば良いでしょうか？各部活においては希望通りその教室を「利用できるか」と「利用できないか」の 2 通りなので，部活の数を n とすると場合の数は 2^n 通りあり，n が 10 以上の場合には 1024 通り以上の組合せになります．この問題に対して，もし各場合に対して選択された部活がそれぞれ両立するかどうか人手で確かめて最大の部活数を求めるとすると大変な作業です．

また，他にも組合せ最適化問題の例として以下の身近な問題が挙げられます：

- 投資問題：一定の資金のもとで，いくつかの株を買って，収益を最大化する問題.
- カーナビのルート探索：二つ地点間の最短（あるいは最安）の行き方を求める問題.
- 修学旅行の観光問題：いくつかの観光地を訪問したいとき，移動時間が最短となる回り方を求める問題.
- 配送計画問題：最短の時間で宅配便を全員に配達するルートを求める問題.
- 仕事場の勤務表の作成：すべての従業員の勤務時間等の希望を満たしつつ勤務表を求める問題.

幅広い分野において多くの現実問題が組合せ最適化問題として捉えることができます．これらの現実問題を組合せ最適化問題として解決するため，まずは問題を数学的な記号を用いて記述します．このことを定式化と呼びます.

高校の教科書では，「領域における最大・最小」問題が紹介されています．これは，一次不等式で表される領域内で一次関数の値を最大化（または最小化）する問題で，**線形計画問題**（Linear Programming, LP）といいます．この線形計画問題を拡張し，その領域内でさらに整数値に限定した問題を**整数計画問題**（Integer Programming, IP）と呼びます.

近年は整数計画問題を解くことができる数理計画ソルバが開発されており [3]，例えば Python 言語のライブラリ PuLP [2] や Gurobi [1] などは簡単に試すことができます．本稿では，いくつかの身近な例を通じて組合せ最適化問題とその整数計画問題への定式化を紹介した後，Python によるソルバの利用方法を説明します.

2　線形計画問題と整数計画問題

高校数学の「領域における最大・最小」問題（線形計画問題）について，以下の問題を考えましょう.

線形計画問題

x，y が 5 つの不等式

$$\begin{aligned} &20x + 15y \leq 3000,\\ &6x + 10y \leq 1200,\\ &12x \leq 1500,\\ &x \geq 0,\\ &y \geq 0 \end{aligned}$$

を満たすとき，$160x + 80y$ のとる値の最大値はいくらでしょうか？

与えられた不等式を満たすような領域は図 1 の水色の領域となります．この領域において $160x + 80y$ の最大値を求めます．このとき，「$160x + 80y$」を線形計画問題における目的関数といいます．$160x + 80y = z$ とし z の値を動かしたとき，赤い点線 $160x + 80y = z$ が水色の領域と共有点を持つ z の中で，z が最大となる点が最適解です．したがって，$x = 125$ と $y = 100/3$ のときに $z = 68000/3$ が最大となり，$160x + 80y$ の最大値は $68000/3$ と分かります.

次に現実生活での応用として，以下の文化祭での生産計画問題を考えましょう.

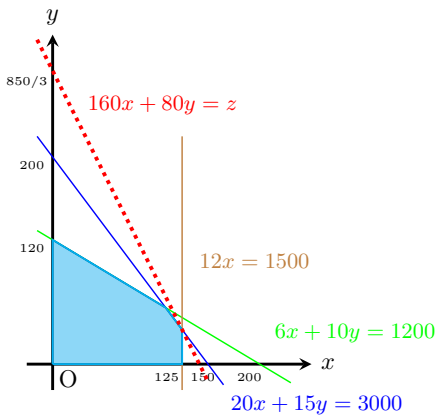

図 1　線形計画問題の解の領域

豆大福と豆餅の生産問題

文化祭で豆大福と豆餅を作って販売します．表 1 に示すように，豆大福と豆餅をそれぞれ一つ作るのに必要な材料が与えられています．豆大福の価格は 160 円，豆餅の価格は 80 円とします．材料として，餅を 3000g，豆を 1200g，あんこを 1500g 準備しています．これらを使って，豆大福と豆餅を生産します．豆大福と豆餅が好きな人はたくさんいるので，売れ残ることはないものとします．豆大福と豆餅をそれぞれいくつ生産すると売り上げが最大になるでしょうか？

表 1　豆大福と豆餅の生産材料

	餅 (g)	豆 (g)	あんこ (g)
豆大福	20	6	12
豆餅	15	10	-

生産する豆大福の個数を x，豆餅の個数を y とします．総売り上げは $z = 160x + 80y$ なので，以下の不等式を満たすとき z のとる値の最大値を求める問題となります．

$$
\begin{aligned}
&20x + 15y \leq 3000,\\
&6x + 10y \leq 1200,\\
&12x \leq 1500,\\
&x,\ y \text{ は非負の整数.}
\end{aligned}
$$

この問題は前に紹介した線形計画問題と最初の三つの不等式は同じですが，x と y の取りうる範囲は図 1 に表す水色の領域の中で値が整数になる部分のみになります．先ほどの線形計画問題と同様に直線を移動する方法で最適解を探すことができます．

この生産計画問題のように，現実の問題は個数のような離散的に表現できる解を求める問題が多いです．次の節ではいくつかの身近な組合せ最適化問題の定式化を紹介します．

3　身近な組合せ最適化

身近な組合せ最適化問題を解くときには，まず問題で与えられたデータとそこから決定したいものを分析します．決定したいものを変数として設定したら，入力データ（定数といいます）と変数を使って，制約条件と目的関数を記述します．これから，生活によく現れる問題の例として，教室予約問題，投資問題，最短ルート探索問題，巡回路探索問題を紹介します．また，それらの問題を整数計画問題として定式化する手法を説明します．

3.1　教室予約問題

まず以下の教室予約問題を定式化してみましょう．

教室予約問題

ある高校で n 個の部活 $N = \{1, 2, \ldots, n\}$ がある教室を利用した活動を希望しています．各部活 $i \in N$ は教室を利用したい開始時刻 s_i と終了時刻 $f_i (> s_i)$ があります．任意の二つ部活 $i, j \in N$ について，$s_i \geq f_j$ あるいは $f_i \leq s_j$（一つの不等式で書くと $\min\{f_i, f_j\} \leq \max\{s_i, s_j\}$）ならば，部活 i と j は両立できるといいます．お互いに両立できる部活の組合せの中で最も部活の数が多い組合せを求めてください．

教室予約問題に対して，定数は部活の集合 N と各部活 i が希望する開始時刻 s_i と終了時刻 f_i です．求めたいのは各部活 i を選択するかどうかですので，ここで，各部活 i を選択するかどうかを表す変数 x_i を作ります．部活 i を選択するならば $x_i = 1$，そうでなければ $x_i = 0$ とします．

制約条件は，もし部活 i と部活 j が両立できない場合，つまり，$\min\{f_i, f_j\} > \max\{s_i, s_j\}$ の場合に，x_i と x_j は同時に 1 になれないというものです．これは以下の式で表せます．

$$x_i + x_j \leq 1, \quad i, j \in N, \min\{f_i, f_j\} > \max\{s_i, s_j\}.$$

目的関数は選択された部活の総数 $\sum_{i \in N} x_i$ で，この値の最大化が目的です．以上より，教室予約問題は以下の整数計画問題に定式化できます．

$$
\begin{aligned}
&\text{最大化} && \sum_{i \in N} x_i\\
&\text{条件} && x_i + x_j \leq 1,\\
& && \quad i, j \in N, \min\{f_i, f_j\} > \max\{s_i, s_j\},\\
& && x_i \in \{0, 1\}, \quad i \in N.
\end{aligned}
$$

3.2 投資問題

次に一定の資金を持つときに，いくつかの株を買うことで収益を最大化する投資問題を考えましょう．

投資問題

n 種類の株 $N = \{1, 2, \ldots, n\}$ と資金額 B が与えられます．株 i の単価は c_i，期待する収益は p_i です．期待収益の合計を最大にする株の買い方を求めてください．

求めたいのは株の買い方ですので，変数 x_i を株 $i \in N$ を買う株数とします．制約条件は，購入額の総額が資金額 B を超えないことなので，以下の式で表せます．

$$\sum_{i \in N} c_i x_i \leq B.$$

目的は購入する株の期待収益の合計 $\sum_{i \in N} p_i x_i$ の最大化です．以上より，投資問題は以下の整数計画問題として定式化できます．

$$\begin{aligned} &\text{最大化} && \sum_{i \in N} p_i x_i \\ &\text{条件} && \sum_{i \in N} c_i x_i \leq B, \\ & && x_i \in \{0, 1, 2, \ldots\}, \quad i \in N. \end{aligned}$$

実は，この問題は**ナップサック問題**と呼ばれる代表的な組合せ最適化問題です．

3.3 最短ルート探索問題

カーナビのルート探索は二地点間の最短（あるいは最安）の行き方を求める問題です．日常生活でよく現れる問題です．

最短ルート探索問題

n 個の地点 $V = \{s, t, 1, 2, \ldots, n-2\}$，$V$ の二地点を結ぶ何本の道の集合 E と各道 $(i, j) \in E$ の長さ $d_{ij} (> 0)$ が与えられます．地点の系列 $(v_1, v_2, \ldots, v_p)$ が $(v_i, v_{i+1}) \in E, i = 1, 2, \ldots, p-1$ を満たすとき地点 v_1 から v_p へのパスと呼びます．パスに含まれる道の長さの合計をパスの長さとします．始点 s から終点 t へのパスの中で長さが最短となるものを求めてください．

図 2 は最短ルート探索問題の一例です．6 個の地点が与えられ，二地点間の道が矢印で表されています．道の長さは矢印の上に書いてあります．この例では，始点 s から終点 t へのパスは (s, t)，$(s, 1, 3, t)$，$(s, 1, 4, t)$ と $(s, 1, 4, 2, t)$ の四つがあります．この四つのパスの中で長さが最短となるものは $(s, 1, 4, 2, t)$ で，その長さは 9 であることが分かります．

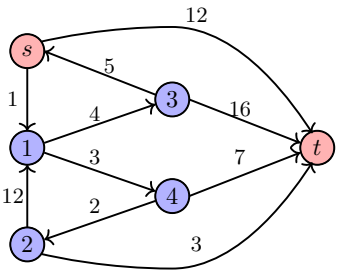

図 2　最短ルート探索問題の例

定式化するにあたり，以下の二つ記号を使います．$O(i)$ を地点 i から出る道を通って次にたどり着く地点の集合，$I(i)$ を地点 i へ行く道がある地点の集合とします．

各道 $(i, j) \in E$ に対して変数 x_{ij} を作り，道 (i, j) を使用するかどうかを表します．道 (i, j) が s から t への最短パスに含まれるならば $x_{ij} = 1$，そうでなければ $x_{ij} = 0$ とします．制約条件の一つは s から出発することです．つまり，s から出発し $O(s)$ のうちのちょうど一つの地点へ移動しないといけません．この条件は以下の式で表せます．

$$\sum_{i \in O(s)} x_{si} = 1.$$

同様に，t に到着する制約は以下の式で表せます．

$$\sum_{i \in I(t)} x_{it} = 1.$$

そのほかのすべての地点 $i \in V \backslash \{s, t\}$ については，i へ行く道と i から出る道の数は等しくならないといけません．ここで，記号「$\backslash$」は集合の引き算を表します．例えば，2 つの集合 A と B に対して，$A \backslash B$ は A に含まれていて B には含まれていない要素の集合を表します．つまり，最短パスで経由する地点 i では出る道と入る道をちょうど一本ずつ選び，それ以外の地点ではいずれの道も選びません．この制約は以下の式で表せます．

$$\sum_{j \in I(i)} x_{ji} - \sum_{k \in O(i)} x_{ik} = 0, \quad i \in V \backslash \{s, t\}.$$

目的関数は使用した道の長さの合計 $\sum_{(i,j) \in E} d_{ij} x_{ij}$ で，この値を最小化することが目的です．最短ルート探索問題は以下の整数計画問題として定式化できます．

$$
\begin{aligned}
&\text{最小化} && \sum_{(i,j)\in E} d_{ij}x_{ij} \\
&\text{条件} && \sum_{i\in O(s)} x_{si} = 1, \\
& && \sum_{i\in I(t)} x_{it} = 1, \\
& && \sum_{j\in I(i)} x_{ji} - \sum_{k\in O(i)} x_{ik} = 0, \quad i \in V\backslash\{s,t\}, \\
& && x_{i,j} \in \{0,1\}, \quad (i,j) \in E.
\end{aligned}
$$

この問題は**最短路問題**と呼ばれる組合せ最適化問題です．

3.4 修学旅行の巡回路探索問題

以下の修学旅行の問題を考えましょう．修学旅行で京都に行くことになりました．A さんは京都のいくつかのお寺と神社で御朱印を拝受することを考えています．A さんはどの順番でお寺と神社を訪問すると移動時間が最短になるでしょうか？

修学旅行の問題

n 個の観光地 $V = \{1, 2, \ldots, n\}$ と任意の二つの観光地 $i, j(\in V)$ の間の移動距離 $d_{ij}(> 0)$ が与えられます．すべての観光地をちょうど 1 回ずつ訪問したあとに出発した場所に戻る経路を巡回路と呼びます．移動距離の合計を最小にする巡回路を求めてください．

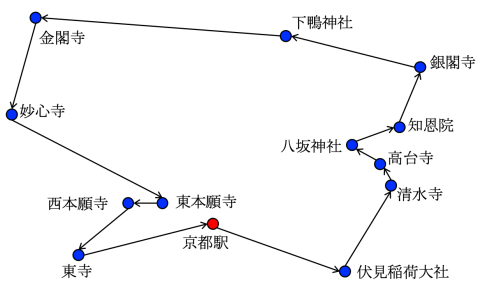

図 3　修学旅行の問題の例

図 3 は修学旅行の巡回路探索問題の一例です．京都駅から出発し，12 ヶ所のお寺と神社を訪問した後でまた京都駅に戻る巡回路を表しています．

各二地点 i と j に対して変数 x_{ij} を作り，地点 i の次に地点 j を訪問するならば $x_{ij} = 1$，そうでなければ $x_{ij} = 0$ します．制約条件はすべての地点をちょうど 1 回ずつ訪問することです．つまり，任意の地点 j に対して，j へ行く道と j から出る道をちょうど一本づつ選びます．この制約は以下の二つの式で表せます．

$$
\sum_{i\in V, i\neq j} x_{ij} = 1, \quad j \in V,
$$

$$
\sum_{i\in V, i\neq j} x_{ji} = 1, \quad j \in V.
$$

しかし，これらの式のみでは図 4 のように分離した複数の部分巡回路からなる解も許してしまいます．図 4 の中で赤の枠で囲まれた部分をよく見ると，一つの部分巡回路の地点の集合 W に対して，部分巡回路に使用した道の数は地点の数 $|W|$ と等しいことがわかります．二つ以上の地点を含む全ての地点の集合 $W \subsetneqq V (|W| \geq 2)$ に対して，使用した道の数は $|W| - 1$ 以下に制限すれば，部分巡回路を禁止することができます．これは以下の式で表せます．

$$
\sum_{i,j\in W} x_{ij} \leq |W| - 1, W \subsetneqq V, |W| \geq 2.
$$

この式は部分巡回路除去不等式といいます．

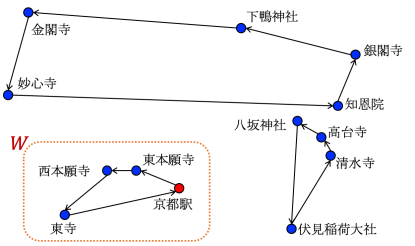

図 4　部分巡回路の例

目的関数は使用した道の距離の合計 $\sum_{i\in V}\sum_{j\in V, i\neq j} d_{ij}x_{ij}$ であり，この値を最小化することが目的です．以上より，修学旅行の問題は以下の整数計画問題に定式化できます．

$$
\begin{aligned}
&\text{最小化} && \sum_{i\in V}\sum_{j\in V, i\neq j} d_{ij}x_{ij} && (1) \\
&\text{条件} && \sum_{i\in V, i\neq j} x_{ij} = 1, \quad j \in V, && (2) \\
& && \sum_{i\in V, i\neq j} x_{ji} = 1, \quad j \in V, && (3) \\
& && \sum_{i,j\in W} x_{ij} \leq |W| - 1, W \subsetneqq V, |W| \geq 2, && (4) \\
& && x_{ij} \in \{0,1\}, \quad i, j \in V, i \neq j. && (5)
\end{aligned}
$$

この問題は**巡回セールスマン問題**と呼ばれる組合せ最適化問題です．

4 数理計画ソルバにより解く

この節では巡回セールスマン問題を例として整数計画問題を数理計画ソルバを使って解く方法を紹介します．ここでは数理計画ソルバとして Python から簡単に使える PuLP を使用します．もしより大規模な問題を解いてみたい方は商用ソルバの Gurobi(アカデミッ

ク版は無料) をお勧めします．紙面の都合上，Python や PuLP の詳細については割愛します．興味のある人は文献 [4] を参照してください．

準備として，まず Python のプログラミング環境を構築します．Python には便利なライブラリが数多く存在しますが，それらをまとめたパッケージである anaconda が `https://www.anaconda.com/products/distribution` で公開されているので，これをインストールします．次に PuLP については，ターミナルで「`pip install pulp`」というコマンドを実行すればインストールができます．

これでプログラミング環境の準備ができたので，巡回セールスマン問題の入力データを設定します．

ソースコード 1　巡回セールスマン問題の入力データ

```
V = {'A':(34, 43),'B': (60, 80),'C': (20, 88),
 'D':(94, 4), 'E':(60, 21), 'F':(17, 57),
 'G': (3, 67), 'H': (12, 41), 'I': (69, 25),
 'J': (90, 97)}
```

ソースコード 1 では，1 行目から 4 行目で都市 A から J までの 10 都市の 2 次元座標を設定しています (図 5)．

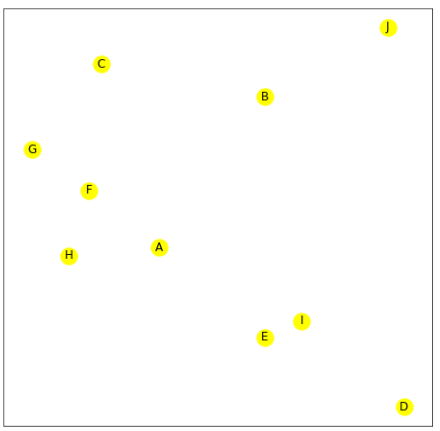

図 5　巡回セールスマン問題の都市

今回の計算では 2 都市間の距離は次のように直線距離とします．ソースコード 2 に 2 都市間の距離を計算する関数 d を示します．

ソースコード 2　距離 d の定義

```
def d(pa,pb):
 return ((pa[0]-pb[0])**2+(pa[1]-pb[1])**2)**0.5
```

巡回セールスマン問題の定式化 (式 (1) 式から式 (5)) を PuLP で表現したものをソースコード 3 に示します．ソースコードの 6 行目と 7 行目で変数 x を定義しています．8 行目では目的関数である式 (1)，10, 11 行目は制約条件 (2)，13, 14 行目は制約条件 (3)，16 から 19 行目は制約条件 (4) をそれぞれ記述しています．そして，21 行目でソルバを実行しています．今回は 10 都市という小さいデータだったので，一瞬で計算が終了します．

ソースコード 3　巡回セールスマン問題に対する整数計画問題

```
import pulp
import itertools

m = pulp.LpProblem('tsp',pulp.LpMinimize)

x ={(i,j):pulp.LpVariable(i+"_"+j,cat='Binary')\
     for i in V for j in V if i != j}
m += sum([d(V[i],V[j])*x[i,j] for (i,j) in x])

for j in V:
    m += sum([x[i,j] for i in V if i != j]) == 1

for j in V:
    m += sum([x[j,i] for i in V if i != j]) == 1

for k in range(2,len(V)):
 for W in itertools.combinations(V,k):
  m+=sum(x[i,j] for i in W for j in W if i!=j)\
      <= len(W)-1

m.solve()
```

ソースコード 4　解の描画

```
import networkx as nx

G=nx.Graph()

for (i,j) in x:
    if x[i,j].value() > 0.99:
        G.add_edge(i,j)

import matplotlib.pyplot as plt
plt.figure(figsize=(8,8))
plt.axis('equal')
nx.draw_networkx(G,pos=V,node_color="yellow")
plt.show()
```

以上の 4 つのソースコードで整数計画問題として定式化された巡回セールスマン問題を解くことができました．そのほかの問題に対しても，同様に簡単に書けますのでぜひ試してみてください．

5　数理計画ソルバは万能？

組合せ最適化問題の解は集合，順序，割当，グラフ，論理値，整数などで表されます．そして，多くの組合せ最適化問題は整数計画問題として定式化できます．では，数理計画ソルバはすべての組合せ最適化問題を実用的な時間で解けるでしょうか？その答えは「NO」です．

日常生活の身近な問題については問題の規模が小さいので，整数計画問題として定式化して数理計画ソルバに渡せば短時間で最適な解を得られる場合が多いでしょう．しかし，問題の性質や規模によって，数理計画ソルバは最適な解を計算できない場合もあります．

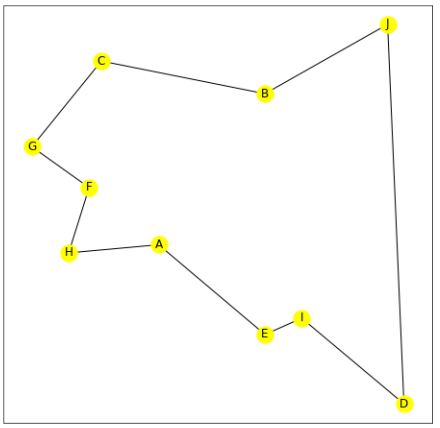

図 6　得られた最適な巡回路

数理計画ソルバを実行して，その計算がしばらく待っても終わらないような状況に直面した場合は，以下の二つの工夫を試してください．

- 実数最適解から近似する．
  整数計画問題は線形計画問題に整数条件を付加した問題ですが，線形計画問題の方がはるかに解きやすいです．現実問題において，安易に整数変数を用いるべきではありません．
  例えば，前述した豆大福と豆餅の生産計画問題については，工場で大量生産する場合には，この問題を整数計画問題に定式化することは必ずしも適切ではありません．この場合は，各変数の整数条件を取り除き，線形計画問題を解くことで実数の最適解を容易に得られます．得られた実数の解の端数を丸めて，最も近い整数解を作れば十分に実用的な解になることが多いです．
- 定式化を見直す
  整数計画問題に定式化するときに，制約条件をどのように数式で表現するかは計算効率に大きな影響を与えます．定式化する際には工夫をすることが必要です．
  例えば，巡回セールスマン問題の定式化において，部分巡回路を持たないようにするために V のすべてのサイズが 2 以上となる部分集合 W に対して，部分巡回路除去不等式

$$\sum_{i,j\in W} x_{ij} \leq |W| - 1, W \subsetneqq V, |W| \geq 2$$

  を課しますが，この不等式の数は非常に多くなってしまいます．訪問したい地点が多い場合は，不等式をソルバに入力することすら難しいです．ランダムに生成する地点集合に対して，筆者の PC（CPU: 4GHz Inter Core i7, メモリ：32GB）で実行した結果を表 2 に示します．表 2 の「単純な定式化」の列がソルバの計算にかかった時間です．不等式の数は地点の数により指数的に増加するため，計算時間も急増します．22 地点の場合は 7200 秒かけても計算が終了しませんでした．

表 2　巡回セールスマンに対する実験結果

	計算時間（秒）	
地点の数	単純な定式化	工夫した定式化
10	0.31	1.09
12	2.07	1.30
15	31.59	1.65
17	199.00	1.54
20	2936.07	17.62
22	-	32.97
30	-	266.81
40	-	1158.58
45	-	5898.15

部分巡回路を禁止するほかの方法を考えなければなりません．一つの方法は，適当な都市 $s \in V$ を選んで出発点とし，巡回路において各都市 i を訪問する順番を表す変数 y_i を作ります．そして，以下の三つの式を課すことで，部分巡回路を禁止できます．

$$\begin{aligned} &y_i - y_j + nx_{ij} \leq n-1, \quad i,j \in V\backslash\{s\}, i \neq j, \\ &y_s = 0, \\ &1 \leq y_i \leq n-1, \quad i \in V\backslash\{s\}. \end{aligned}$$

この三つの式を巡回セールスマン問題の定式化の式 (4) と入れ替えた新しい定式化で解いた結果を表 2 の「工夫した定式化」の列に示します．40 地点までには現実的な時間で最適な巡回路を計算できました．45 地点を含む例に対して，最適なルートを計算するために 5898.15 秒かかりました．一概に言えませんが，PuLP を使う場合に実用的な時間（例えば 7200 秒以内）で計算できる地点の数は 50 程度でしょう．

上記のように，巡回セールスマン問題に対しては定式化を工夫しても 6000 秒以内で計算できた地点の数は 45 までででした．実は数理計画ソルバが苦手なタイプの問題があります．例えば，図形を容器に効率よく詰め込む配置問題や巡回セールスマン問題のような順序を求める問題に対しては，問題の規模が少し大

きくなると，どのように定式化を工夫しても現時点では数理計画ソルバで良質な解を得ることは難しいでしょう．

工学的な応用においては，身近な問題より問題の規模がはるかに大きいです．また，組合せ最適化問題の中には本質的に解くことが困難なものが多いので，整数計画問題に定式化できたからといって数理計画ソルバで簡単に解けるわけではないです．そのような問題を解きたいときには，問題の性質にあわせて専用のアルゴリズムを設計しないといけません．さらに深く勉強したいなら，文献 [5, 6] を推薦します．

6 おわりに

本稿では身近な問題の整数計画問題への定式化と数理最適化ソルバを紹介しました．日常生活で気をつけていると，身近なたくさんの問題が最適化問題の性質を持つことがわかります．これらの問題を定式化し，数理最適化ソルバで解くことはとても楽しいです．定式化を工夫したり，できるだけ良い解をできるだけ短い時間で求めるアルゴリズムを探求することも研究の醍醐味です．本稿を通して，応用数学や数理最適化に少しでも興味を持ってもらえたら嬉しいです．

参考文献

[1] Gurobi optimization. https://www.gurobi.com/. 閲覧日 2023 年 1 月 5 日.

[2] Optimization with PuLP. https://coin-or.github.io/pulp/. 閲覧日 2023 年 1 月 5 日.

[3] 宮代隆平. 整数計画ソルバー入門. オペレーションズ・リサーチ, Vol. 57, No. 4, pp. 183–189, 2012.

[4] 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.

[5] 梅谷俊治. しっかり学ぶ数理最適化. 講談社, 2020.

[6] 柳浦睦憲, 茨木俊秀. 組合せ最適化—メタ戦略を中心として. 朝倉書店, 2001.