都市情報システム実習（檀担当分）　[第 4 週]

1　本日の実習テーマ

- 巡回セールスマン問題 (Traveling Salesman Problem: TSP)
  - 前回（第 3 週）で扱った解法とは異なる手法で TSP を解いてみる

2　巡回セールスマン問題の様々な解法

2.1　問題の定義（再掲）

巡回セールスマン問題 (Traveling Salesman Problem: TSP) とは次のような問題です：

あなたは某社の有能セールスマンです．今回，新商品の発売のため，n 箇所を訪問することになりました．有能なあなたは,「本社を出発して，できるだけ短い移動距離で全ての訪問場所を回り，また本社に帰ってこよう」と考えました．手元には，全ての訪問箇所間（本社含む）の距離一覧があります．さて，どのような経路で全ての訪問場所を回ればいいか，計画を立ててください．

2.2　前回のモデル

前回は，TSP を解くためのモデルとして次のようなものを考えました：

【集合・要素】

- $i \in I := \{1, 2, \ldots, n\}$, $j \in J := \{1, 2, \ldots, n\}$: 訪問箇所の集合（本社 (= スタート地点: 1 をスタート地点とする) 含む）
  - 「訪問箇所の集合」に対して 2 通りの表現方法 ($i \in I, j \in J$) を準備する．訪問順序の前後を考慮するときには $i \in I$（前）,$j \in J$（後）を区別する．

【パラメータ】

- n: 訪問箇所数（本社含む）
- d_{ij}: 訪問箇所 i と j の間の距離

【変数】

$$x_{ij} \ := \ \begin{cases} 1, & \text{訪問箇所 } i \text{ の次に } j \text{ を訪問する} \\ 0, & \text{それ以外} \end{cases}$$

$$u_i \ := \ \text{訪問箇所 } i \text{ の訪問順番 (本社を 0 番目とする)}$$

$$\begin{array}{|ll} \text{minimize} & \displaystyle\sum_{i \in I, j \in J} d_{ij} x_{ij} \\ \text{subject to} & \displaystyle\sum_{j \in J} x_{ij} = 1 \quad (i \in I) \\ & \displaystyle\sum_{i \in I} x_{ij} = 1 \quad (j \in J) \\ & x_{ij} = 0 \quad (i \in I, j \in J \text{ s.t. } i = j) \\ & u_i + 1 - (n-1)(1 - x_{ij}) \le u_j \\ & \qquad (i \in I, j \in J \text{ s.t. } i \ge 2, j \ge 2, i \ne j) \\ & 1 \le u_i \le n - 1 \quad (i \in I \text{ s.t. } i \ge 2) \\ & x_{ij} \in \{0, 1\} \quad (i \in I, j \in J) \end{array} \tag{1}$$

このモデルの特徴は，u_i という，訪問順番を表す変数を準備しているところです．この変数が所定の意味を持つように，やや複雑な制約（(1) で u_i が現れる制約）を準備しています．この制約の意味については，前回資料を参照してください．

2.3　違うモデルを考えてみる

本日の実習では，前回とは異なる解法で TSP を解いてみたいと思います．

まず，以下のモデルを見て下さい：

$$\begin{array}{|ll} \text{minimize} & \displaystyle\sum_{i \in I, j \in J} d_{ij} x_{ij} \\ \text{subject to} & \displaystyle\sum_{j \in J} x_{ij} = 1 \quad (i \in I) \\ & \displaystyle\sum_{i \in I} x_{ij} = 1 \quad (j \in J) \\ & x_{ij} = 0 \quad (i \in I, j \in J \text{ s.t. } i = j) \\ & x_{ij} + x_{ji} \le 1 \quad (i \in I, j \in J) \\ & x_{ij} \in \{0, 1\} \quad (i \in I, j \in J) \end{array} \tag{2}$$

前回のモデル (1) からいくつかの制約（特に u_i に関する制約）を取り除き，新たに $x_{ij} + x_{ji} \le 1$ $(i \in I, j \in J)$ という制約が追加されたものになっています．追加された制約は「x_{ij} と x_{ji} が同時に 1 にはならない」ということですから,「訪問箇所 i から j に行くならば，j から i には行かない」という意味になります．

このモデルを用いて問題を解いてみると（データは前回資料の 0.dat を利用），以下のような回答を得ます（解ファイルからの抜粋）:

```
Objective:  Objective = 393.92 (MINimum)
      3 x[1,3]     *   1   0   1
     32 x[2,12]    *   1   0   1
     60 x[3,20]    *   1   0   1
     74 x[4,14]    *   1   0   1
     86 x[5,6]     *   1   0   1
    102 x[6,2]     *   1   0   1
    128 x[7,8]     *   1   0   1
    145 x[8,5]     *   1   0   1
    178 x[9,18]    *   1   0   1
    197 x[10,17]   *   1   0   1
    210 x[11,10]   *   1   0   1
    221 x[12,1]    *   1   0   1
    249 x[13,9]    *   1   0   1
    276 x[14,16]   *   1   0   1
    287 x[15,7]    *   1   0   1
    304 x[16,4]    *   1   0   1
    335 x[17,15]   *   1   0   1
    353 x[18,13]   *   1   0   1
    371 x[19,11]   *   1   0   1
    399 x[20,19]   *   1   0   1
```

一方，前回のモデル (1) を用いて解いた結果は次のようになります：

```
Objective:  Objective = 404.73 (MINimum)
      3 x[1,3]     *   1   0   1
     32 x[2,12]    *   1   0   1
     60 x[3,20]    *   1   0   1
     74 x[4,14]    *   1   0   1
     86 x[5,6]     *   1   0   1
    102 x[6,2]     *   1   0   1
    128 x[7,8]     *   1   0   1
    144 x[8,4]     *   1   0   1
    177 x[9,17]    *   1   0   1
    198 x[10,18]   *   1   0   1
    210 x[11,10]   *   1   0   1
    221 x[12,1]    *   1   0   1
    249 x[13,9]    *   1   0   1
    276 x[14,16]   *   1   0   1
    287 x[15,7]    *   1   0   1
    305 x[16,5]    *   1   0   1
    335 x[17,15]   *   1   0   1
    353 x[18,13]   *   1   0   1
    371 x[19,11]   *   1   0   1
    399 x[20,19]   *   1   0   1
```

…どうやら，答えが違うようです．特に，(1) で得た最短巡回路の長さが 404.73 であるのに対し，(2) で得た長さは 393.92 となっています．前回のモデルが間違っているのでしょうか？今回のモデルが間違っているのでしょうか？

それを調べるために，今回のモデル (2) で得た解をよく見てみましょう．すると，解が次の 3 つの部分に分かれることがわかります．

```
      3 x[1,3]     *   1   0   1
     60 x[3,20]    *   1   0   1
    399 x[20,19]   *   1   0   1
    371 x[19,11]   *   1   0   1
    210 x[11,10]   *   1   0   1
    197 x[10,17]   *   1   0   1
    335 x[17,15]   *   1   0   1
    287 x[15,7]    *   1   0   1
    128 x[7,8]     *   1   0   1
    145 x[8,5]     *   1   0   1
     86 x[5,6]     *   1   0   1
    102 x[6,2]     *   1   0   1
     32 x[2,12]    *   1   0   1
    221 x[12,1]    *   1   0   1
--------------------------------
     74 x[4,14]    *   1   0   1
    276 x[14,16]   *   1   0   1
    304 x[16,4]    *   1   0   1
--------------------------------
    178 x[9,18]    *   1   0   1
    353 x[18,13]   *   1   0   1
    249 x[13,9]    *   1   0   1
```

つまり,「すべての訪問箇所を通ってはいるが，単一の巡回路は得られていない」ということになります．図にすると図 2 のようになります（図 1 と比較してみてください）:

2.4 部分巡回路除去制約 [1]

図 2 からわかるように，巡回路が分割されていることがわかります．これを「部分巡回路」といいます．

前回のモデル (1) では，訪問順番を制限する u_i という変数があったため，部分巡回路は発生しませんでした．一方，モデル (2) では，2 回の移動で発生する部分巡回路については（今回追加した $x_{ij}+x_{ji} \leq 1$ $(i \in I, j \in J)$ という制約で）制限していますが，3 回以上の移動で発生する部分巡回路については制限していません．

このような部分巡回路を一律に制限するためには，全ての部分巡回路 を列挙し，それだけ制約を追加する必要があります．しかし,『全て』を列挙するのは無謀です！例えば，3 回の移動で発生する部分巡回路の数は ${}_nC_3$ 通り（訪問箇所数が n の場合），4 回の移動で発生する部分巡回路の数は ${}_nC_4 \times 4!/4 ={}_n C_4 \times 6$ 通り（4 箇所を巡る巡回路は 4!/4 通りある），…と爆発的に増えていきます．

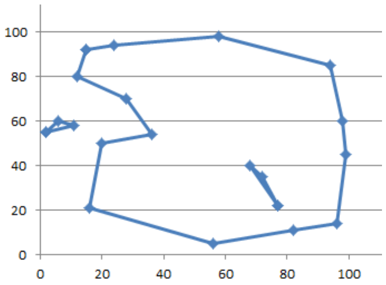

図 1: (2) による解

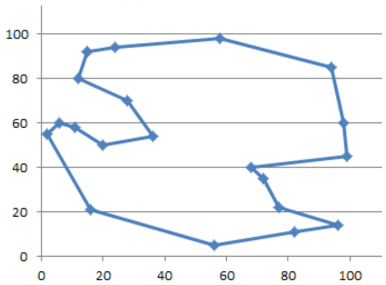

図 2: (1) による解

しかし，仮に部分巡回路を制限せずにモデル (2) を解き，単一の巡回路が得られれば，それは最適な巡回路であるということが言えます.『制約がなくとも最短の巡回路として単一の巡回路が得られた』のですから,『部分巡回路を制限するような制約を加えても，得られる単一の巡回路は同じものになる』ということです.

これを発展させると，以下のような戦略が可能になることがわかるでしょう：

- もし部分巡回路が発生したら，その部分巡回路を制限するような制約を追加した問題を解く.
- 以下それを繰り返し，どこかで単一の巡回路が得られれば，それが最適解である.

今回はこの戦略に基づいて TSP を解いてみたいと思います.

部分巡回路を制限するような制約を追加した問題は以下のように定義されます[1]：

$$
\left|
\begin{array}{ll}
\text{minimize} & \displaystyle\sum_{i\in I, j\in J} d_{ij}x_{ij} \\
\text{subject to} & \displaystyle\sum_{j\in J} x_{ij} = 1 \quad (i \in I) \\
 & \displaystyle\sum_{i\in I} x_{ij} = 1 \quad (j \in J) \\
 & x_{ij} = 0 \quad (i \in I, j \in J \text{ s.t. } i = j) \\
 & x_{ij} + x_{ji} \le 1 \quad (i \in I, j \in J) \\
 & x_{ij} \in \{0,1\} \quad (i \in I, j \in J) \\
 & \displaystyle\sum_{i,j\in V_k, i\neq j} x_{ij} \le p_k - 1 \quad (k \in K) \\
 & x_{ij} \in \{0,1\} \quad (i \in I, j \in J)
\end{array}
\right. \tag{3}
$$

ここで，$k \in K$ は部分巡回路の集合になり，V_k は部分巡回路 k に含まれる節点の集合，p_k は部分巡回路 k に含まれる枝の本数になります.

部分順回路 k は節点集合 V_k に枝を p_k 本持つ巡回路です．実はこのような巡回路は $(p_k-1)!$ 通りあります（理由を考えてみてください）．これらの部分順回路は，求める解（= サラリーマンの巡回路）に含まれてはならないので，これらを禁止する必要があります．そのために，

$$\sum_{i,j\in V_k, i\neq j} x_{ij} \le p_k - 1 \quad (k \in K)$$

という制約を設けています．この制約の下では，節点集合 V_k に枝を p_k 本持つ巡回路は発生しないことになります.

これらの制約式に関するモデルファイル･データファイルの書き方についても少し注意が必要になります．まず，$k \in K$ と V_k については次のように書きます：

```
set K;
set V{K};
```

また，制約については

```
subject to C5 {k in K}:
    sum {i in V[k], j in V[k]: i != j}
        x[i, j] <= p[k] - 1;
```

のように書きます（パラメータ p{K} も準備しておく必要がありますね）.

さらにデータファイルについては以下のように書きます（該当部分のみ）:

```
set K := 1 2 3;
set V[1] := 1, 3, 20, 19, 11, 10, 17, 15, 7, 8,
            5, 6, 2, 12;
set V[2] := 4, 14, 16;
```

[1] 再履修の方へ：2014 年度までの資料からモデルを少し変更しています.

```
set V[3] := 9, 18, 13;
param p :=
[1] 14
[2] 3
[3] 3
;
```

<snip> は「中略」の意味です．実際には，解ファイルの内容に応じて適切な内容を書く必要があります．2.3 節の解ファイルの内容とも比較してみてください．

まとめると，今回は次のような方法で TSP を解くことになります．

---

0. (2) を解く．**1.** へ．

1. 単一の巡回路が得られれば計算終了．さもなくば **2.** へ．

2. 部分巡回路に関するデータを追加し (3) を解く．**1.** へ．

0.dat の場合，(3) を 1 回解いた時点で単一の巡回路を得ることができました．場合によっては，(3) を複数回解くこともあるでしょう．

ただ，一回当たりの求解時間は (1) を解くときよりもかなり短くなっていることがわかると思います（`--log` オプションで確認すると良いでしょう）．今回は，部分巡回路に関するデータの追加は手作業で行っていますが，この部分を自動化すると全体としても計算時間を（多くの場合は大きく）短縮できます．実際，TSP を解くときに (1) を用いることは少なく，(2), (3)) を用いることが多いです [2]．

参考文献

[1] 藤江 哲也，最近の混合整数計画ソルバーの進展について，日本オペレーションズ・リサーチ学会 学会誌 2011 年 5 月号．

[2] 前回の実習において (1) で解けない問題に ’当たった’ 方も，(2), (3)) を用いれば解ける可能性はかなりあります．是非 try してみてください．