都市情報システム実習（檀担当分）　[第 3 週]

1　本日の実習テーマ

- 巡回セールスマン問題 (Traveling Salesman Problem: TSP)

TSP を解くことを通じて，

- 最適化モデルに現れる制約条件の意味を読み解く
- 最適化問題の解を分析する

という能力を養います．

2　巡回セールスマン問題

2.1　問題の定義

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　モデル

この問題を解くための最適化モデルは，以下のようになります：
【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】

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

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

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

2.3　制約条件を読み解く

一般に，最適化モデルを組み立てるためには，多くの（良質な）最適化モデルを読み解くことで，自分が使える数学的表現の方法を増やしていく必要があります．ここではその練習を行います．

上記 (1) に現れる制約式のうち，

$$\begin{aligned} & u_i + 1 - (n-1)(1 - x_{ij}) \le u_j \\ & \quad (i \in I, j \in J \text{ s.t. } i \ge 2, j \ge 2, i \ne j) \\ & 1 \le u_i \le n - 1 \quad (i \in I \text{ s.t. } i \ge 2) \end{aligned}$$

に注目しましょう（この制約式は Miller-Tucker-Zemlin (MTZ) 制約 [1] と呼ばれる制約式です）．変数 u_i が現れるのはこの 2 つの制約式ですね．ということは，これらの式を読み解いていけば，u_i の持つ意味がわかってきそうです．

まず $x_{ij} = 0$ の場合，すなわち「訪問箇所 i の次に j を訪問しない」場合を考えてみます．このとき，最初の制約式は

$$u_i + 1 - (n-1) \le u_j \tag{2}$$

となります．一方，$1 \le u_i \le n-1$, $1 \le u_j \le n-1$ です（※ j も訪問箇所を表す要素なので，u_j についても u_i と同じ式が成立する）．制約 (2) を最も「破綻」させそうなケースは，u_i ができるだけ大きく，u_j ができるだけ小さい場合，すなわち $u_i = n-1$, $u_j = 1$ の場合です．しかしその場合でも，(左辺)$= 1 =$(右辺) で

あり，制約式は満たされます．すなわちこの制約式は常に満たされるということになり，$x_{ij} = 0$ の場合は事実上意味を持たない制約になることがわかります．

次に $x_{ij} = 1$ の場合，すなわち「訪問箇所 i の次に j を訪問する」場合を考えます．この場合，ここで考えている制約式は

$$u_i + 1 \leq u_j \tag{3}$$

となります．つまり，$i \to j$ の順に訪問するならば，u_j の値は u_i よりも 1 以上 大きくなる，ということです．このとき，k 番目に通過する訪問箇所を i_k とし，もう一つの制約に注意すると，

$$1 \leq u_{i_1} \leq_1 u_{i_2} \leq_1 \cdots \leq_1 u_{i_{n-1}} \leq n-1$$

という大小関係が成立します（$i \geq 2$ とあるので，スタート地点 $(i = 1)$ は含まないことに注意）．ここで $a \leq_1 b$ は「b は a より 1 以上大きい」ということを表すものとします．そして，最左項 (1) と最右項 $(n-1)$ に挟まれる項の数に注意すれば，結果として

$$u_{i_1} = 1,\ u_{i_2} = 2,\ \ldots,\ u_{i_{n-1}} = n-1$$

となることがわかります．すなわち，u_i は訪問箇所 i の訪問順序を表すものに他なりません．

ただし，ここで考えた制約条件には，スタート地点が含まれていないことに注意してください．これは何故でしょうか？考えてみて下さい．

また，他の制約条件についても，どのような意味を持つものなのか，考えた結果をレポートにまとめてください．

2.4 モデルの記述

実際に問題を解くには，モデルを .mod ファイルに記述する必要がありますね．これまでに記述してきた .mod ファイルを参考にすれば，記述できると思います．

ただし，制約条件の範囲を指定する方法についてはまだ出てきていませんので説明しておきます．例えば (1) の 4 番目の制約式は，

```
subject to C4 {i in I, j in J:
                i >= 2 && j >= 2 && i != j}:
u[i] + 1 - (n - 1) * (1 - x[i, j]) <= u[j];
```

などと記述します．&&, || や !=, == など，Java や C などと同じ条件演算子を利用することができます．

2.5 問題データ

今回は，個人別に問題データを作成してあります．data.zip を展開し，自分の学籍番号が付いた .dat ファイルを利用してください [1]．

今回のデータは，訪問箇所が 20 箇所（スタート地点を含む）のデータを作成しました．ファイルを見て頂くと，*** Point Data *** と *** Distance Data *** という部分があるのがわかると思います．*** Point Data *** では，各訪問箇所が $x-y$ 平面にあるとしたときの x 座標，y 座標を上から順に列挙しています．*** Distance Data *** は，訪問箇所 1〜20 の間の距離を表しています．各行の一番左に訪問箇所 i の番号が書いてあり，右側には i から各訪問箇所への距離が訪問箇所の番号順に記載してあります．

本データを加工することで，glpk で利用するデータファイルを作成してください（※そのままではデータファイルにはなりません！）．

2.6 結果のまとめ方

最適化計算を（いつもと同じように）行い，解ファイルを（-o オプションで）出力します．ここでは解ファイル名を w3p1.sol とします．このとき，コマンドプロンプトで

```
grep -e "x\[.*\] *\* *1" < w3p1.sol
```

（\ は半角の￥）とすると，以下のような表示を得ます [2]：

```
  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
```

これは，x_{ij} が 1 となっている箇所のみを取り出したものになっています．また解ファイルをみると，u_i についても以下のような記載があると思います：

[1] 本講義 2 回目の履修者の方は学籍番号に 100 を足すか引くかした番号のファイルを利用してください．

[2] 紙面の都合上，幅を狭めました．なお，この結果は 0.dat に対する計算結果です．

```
401 u[2]          18
402 u[3]           1
403 u[4]          13
404 u[5]          16
405 u[6]          17
406 u[7]          11
407 u[8]          12
408 u[9]           8
409 u[10]          5
410 u[11]          4
411 u[12]         19
412 u[13]          7
413 u[14]         14
414 u[15]         10
415 u[16]         15
416 u[17]          9
417 u[18]          6
418 u[19]          3
419 u[20]          2
```

これらから，訪問箇所の訪問順序を読み解いて下さい．

最後に，答えを確認する必要がありますね．得られたの巡回路を，Excel などを使って書いてみて下さい．訪問箇所の x 座標，y 座標は [学籍番号].dat に記載がありましたね．

例えば図 1 のようになっていれば概ね正しそうですが，図 2 のような絵ができた場合は何かが間違っていそうですね…

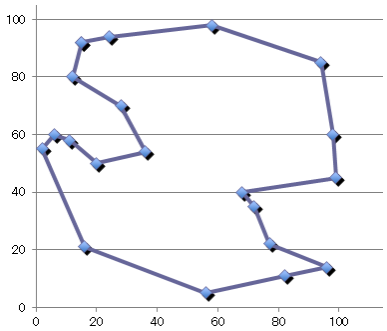

図 1: よい例

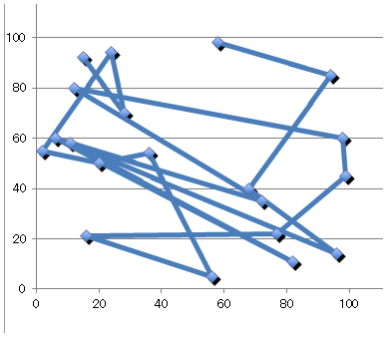

図 2: 悪い例

完成した図は是非レポートに含めて下さい！

参考文献

[1] 久保幹雄，サプライチェイン最適化ハンドブック，朝倉書店 (2007).