APM 523 Mittleman
Project 1
Matt Kinsinger, Hongjun Choi, Adarsh Akkshai
Due 8-30-2017

The optimization problem we're solving is the traveling salesman problem. The problem in itself is $\mathcal{NP}$ hard to solve by brute force and hence requires us to subject certain constraints to optimize the route of travel. Examples of traveling salesman could also include applications in inventory management of lifting crates by cranes, or the distance one needs to optimally place their fingers on a keyboard to reach all keys.

To demonstrate this we chose the smallest set of cities present in the TSPLIB. They are geographical location of the cities in Burma: a set of 14 cities. We now set up the TSP problem as follows.

Given a set of nodes i, j, and distances d_{ij} we need to minimize;

$$\min \sum_{i,j} d_{i,j} x_{i,j}$$
$$\text{subject to : } \sum_{i} x_{i,j} = 1$$
$$\text{subject to: } \sum_{j} x_{i,j} = 1$$

where the nodes $i, j \in \{1, \ldots, 14\}$, as the set of cities, $x_{i,j} \in \{0, 1\}$ denotes the weights of travel from one city to another. We also denote distance from $i \rightarrow j$ as forward distance d_{ij} and d_{ji} as backward distance from $i \leftarrow j$ (To avoid misconceptions we also define that the if the forward and backward distances are identical it is a symmetric TSP problem, else it is a asymmetric TSP problem).
Since the given data set is in geographical co-ordinates (latitude,longitude). We convert them as true distances in kms by:
First converting them into radians;

$\pi \approx 3.141592$
$deg = floor(x(i))$
$min = x(i) - deg$
$latitude(i) = \pi \times (deg + 5.0 * min/3.0)/180.0$
$deg = floor(y(i))$
$min = y(i) - deg$
$longitude(i) = \pi * (deg + 5.0 * min/3.0)/180.0$

Once we have the co-ordinates in radians, we evaluate the distance

$RRR = 6378.388$(Radius of earth in kms)
$q1 = \cos(longitude(i) - longitude(j))$
$q2 = \cos(latitude(i) - latitude(j))$
$q3 = \cos(latitude(i) + latitude(j))$
$d(i, j) = floor(RRR \times \cos(0.5 \times ((1.0 + q1) \times q2 - (1.0 - q1) \times q3)) + 1.0)$

We evaluated the true distances between the cities after applying the transformation and found them to be the distance matrix $\mathbf{D}$:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	inf	153	510	706	966	581	455	70	160	372	157	567	342	398
2	.	inf	422	664	997	598	507	197	311	479	310	581	417	376
3	.	.	inf	289	744	390	437	491	645	880	618	374	455	211
4	.	.	.	inf	491	265	410	664	804	1070	768	259	499	310
5	.	.	.	.	inf	400	514	902	990	1261	947	418	635	636
6	.	.	.	.	.	inf	168	522	634	910	593	19	284	239
7	.	.	.	.	.	.	inf	389	482	757	439	163	124	232
8	.	.	.	.	.	.	.	inf	154	406	133	508	273	355
9	.	.	.	.	.	.	.	.	inf	276	43	623	358	498
10	.	.	.	.	.	.	.	.	.	inf	318	898	633	761
11	.	.	.	.	.	.	.	.	.	.	inf	582	315	464
12	.	.	.	.	.	.	.	.	.	.	.	inf	275	221
13	.	.	.	.	.	.	.	.	.	.	.	.	inf	247
14	.	.	.	.	.	.	.	.	.	.	.	.	.	inf

Table 1: The distance matrix **D**

where $d_{ji} = .$ denotes the symmetric distance d_{ij}.

A point to be noted: Here since we're under the underlying assumption that the distances are symmetric only the upper triangular matrix of **D** has been evaluated, while the $i = j$ distances scaled to ∞. Now that we have the distances evaluated neatly in a matrix form, we proceed with the questions in the assignment.

The co-ordinate data has been saved in **burma14_ TSP_ data.txt**

Matlab code to generate distance matrix: **distance_ matrix.m**

a) The problem demands we choose the symmetric matrix, which means it is sufficient to use the upper triangular matrix of **D**.

   i) Part A

   We then use this distance matrix as data in our submission to NEOS to solve the TSP problem. In the model of tsp.mod, we eliminate subtours by first building the set $SS = \{1, \ldots, 2^n - 2\}$ to be an index set for the set POW= $\mathbb{P}(S) - \{S, \emptyset\}$, where $\mathbb{P}(S)$ is the power set of S. Then POW is built using a clever combinatorical programming scheme that assigns to each element of SS a particular subset of S (excluding S and $\emptyset$). Using this we can eliminate the subtours $C \subseteq$ POW using a constraint:

$$\sum_C x_{i,j} \leq |C| - 1,\ 3 \leq |C| \leq n - 1,\ C \subseteq \text{POW}$$

   Where $x_{i,j} = 1$ if the leg $i \to j$ is included in our tour, and 0 otherwise. This constraint eliminates all subtours of our set S by

   1. Selecting only the sets of cities in POW that have at least 3 cities and no more than cardinality $|S| - 1$ cities.

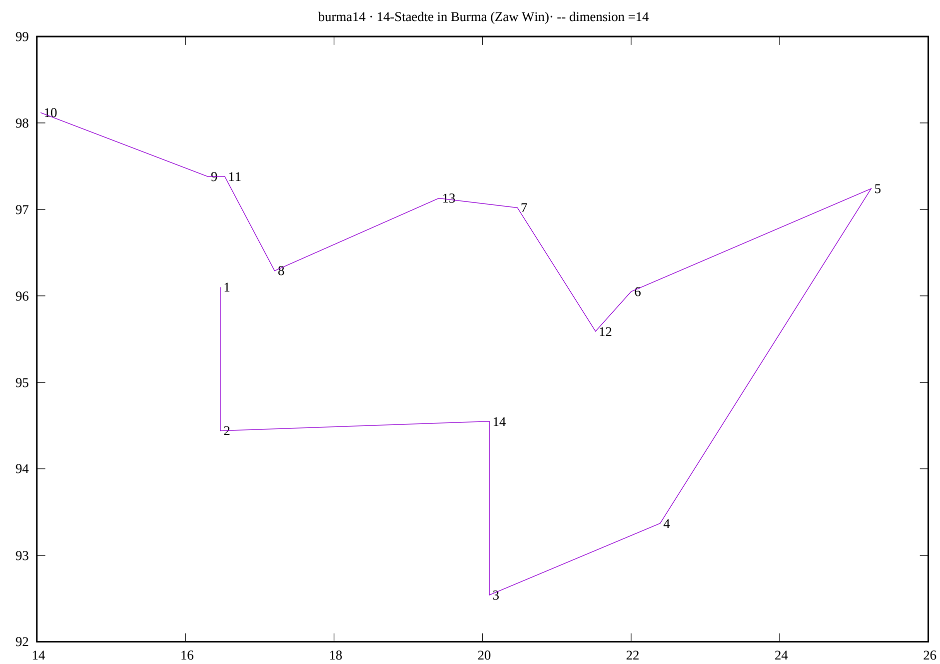
2. For each set C that is selected it forces the sum of the x_{ij} that corresponds to connections between the cities of C to be less than cardinality $|S|-1$. This guarantees that no proper subset $B \subset S$ with cardinality $|B| - 1 \geq 2$ has all of its elements connected by a closed loop. Thus eliminating any subtours.

   The optimal tour was

$$1 \to 2 \to 14 \to 3 \to 4 \to 5 \to 6 \to 12 \to 7 \to 13 \to 8 \to 11 \to 9 \to 10.$$

   The distance traveled on this tour is 3,323 miles, which is the most optimal route possible. We feed the data from burma.tsp into NEOS Concorde solver to solve the TSP problem and print out a picture of the optimal tour. apm523-5 Neos model:

   tsp.mod
   Neos data: **burma14.dat**
   Neos commands: **tsp_ comm.txt**
   Neos tsp.mod results: **burma14_ tsp_ neos_ results.txt**
   Neos Concorde tour printout: **burma14_ map.pdf**



ii) We now use a different method to solve the symmetric TSP problem. In this method we submit our problem to NEOS without the constraints that eliminated all subtours in part i). We then see the resulting answer inevitably contains subtours and after viewing the results we place constraints into the model to eliminate those specific subtours. This process was repeated 6 times until we had eliminated all subtours that the algorithm gave as solutions. Note that "elim1" would eliminate the specific subtour of cities $a \to h \to b \to a$. This code can be adjusted to handle subtours of any size. The iterations are shown below:

subj to elim1: $X['a','h'] + X['a','b'] + X['b','h'] <= 2;$

subj to elim2: $X['c','d'] + X['d','e'] + X['e','f'] + X['f','l'] + X['g','l'] + X['g','m'] + X['m','n'] + X['c','n'] <= 7;$

subj to elim3: $X['i','j'] + X['j','k'] + X['i','k'] <= 2;$

Second set of subtours that we need to eliminate

subj to elim4: $X['a','b'] + X['b','j'] + X['i','j'] + X['i','k'] + X['h','k'] + X['a','h'] <= 5;$

subj to elim5: $X['c','d'] + X['d','e'] + X['e','l'] + X['f','l'] + X['f','g'] + X['g','m'] + X['m','n'] + X['c','n'] <= 7;$

Third set of subtours that we need to eliminate

subj to elim6: $X['a','b'] + X['b','h'] + X['h','k'] + X['i','k'] + X['i','j'] + X['a','j'] <= 5;$

subj to elim7: $X['c','d'] + X['d','e'] + X['e','f'] + X['f','l'] + X['l','m'] + X['g','m'] + X['g','n'] + X['c','n'] <= 7;$

Fourth set of subtours that we need to eliminate

subj to elim8: $X['a','b'] + X['b','j'] + X['j','k'] + X['i','k'] + X['h','i'] + X['a','h'] <= 5;$

subj to elim9: $X['c','d'] + X['d','l'] + X['f','l'] + X['e','f'] + X['e','g'] + X['g','m'] + X['m','n'] + apm523 - 5X['c','n'] <= 7;$

Fifth set of subtours that we need to eliminate

subj to elim10:

$X['a','h'] + X['b','h'] + X['b','j'] + X['i','j'] + X['i','k'] + X['a','k'] <= 5;$

subj to elim11: $X['c','d'] + X['d','e'] + X['e','l'] + X['f','l'] + X['f','m'] + X['g','m'] + X['g','n'] + X['c','n'] <= 7;$

Sixth set of subtours that we need to eliminate

subj to elim12:

$X['a','b'] + X['b','h'] + X['h','j'] + X['i','j'] + X['i','k'] + X['a','k'] <= 5;$

subj to elim13: $X['c','d'] + X['d','f'] + X['f','l'] + X['e','l'] + X['e','g'] + X['g','m'] + X['m','n'] + X['c','n'] <= 7;$

We have placed comments in tsp1.mod to show the sequence of iterations and which subtours we eliminated. We ended up with the same optimal tour and distance as in i). Another important conclusion we came towards is the time required for computations by specific subtour elimination was substantially lower since we didn't have to iterate over all the possible subtours. While this provides lower time, the process is laborious, especially when the size of the cities grows very large.

Neos model: **tsp1.mod**

Neos data: **burma14.dat**

Neos final results: **same as in part i)**

b) Part B

There can be many examples where the TSP problem might lead to asymmetry, say for example when the salesman is moving from one city to another, he might encounter unexpected traffic which increases the distance along one leg of the journey. He could also have issues with roads not leading both ways, meaning he has to take detours to reach the same city. Other examples might include minor asymmetry in phalanges to reach all keys of a keyboard. We compute $\mathbf{D}'_{n\times n}$ as the asymmetric distance matrix where d'_{ij} and d'_{ji} are the new distances (old distance [Burma14 data] $\pm 10\%$) due to the westerly wind being blown (It is different from case i). The asymmetric distance is:

$$\begin{bmatrix}
. & 169 & 561 & 777 & 870 & 639 & 410 & 63 & 144 & 335 & 141 & 623 & 308 & 438 \\
138 & . & 465 & 730 & 897 & 538 & 457 & 177 & 280 & 431 & apm523-5279 & 523 & 375 & 338 \\
459 & 380 & . & 260 & 669 & 351 & 394 & 442 & 580 & 792 & 557 & 336 & 410 & 190 \\
635 & 598 & 317 & . & 442 & 238 & 369 & 598 & 724 & 963 & 691 & 233 & 449 & 279 \\
1063 & 1097 & 818 & 540 & . & 440 & 565 & 992 & 891 & 1135 & 852 & 460 & 699 & 700 \\
523 & 658 & 429 & 291 & 360 & . & 151 & 469 & 571 & 819 & 534 & 20 & 256 & 263 \\
501 & 558 & 481 & 451 & 463 & 185 & . & 428 & 433 & 681 & 395 & 179 & 112 & 256 \\
77 & 216 & 541 & 730 & 811 & 574 & 350 & . & 139 & 366 & 119 & 559 & 246 & 390 \\
176 & 343 & 709 & 885 & 1089 & 698 & 530 & 170 & . & 249 & 43 & 685 & 393 & 548 \\
409 & 527 & 968 & 1177 & 1387 & 1001 & 832 & 447 & 304 & . & 350 & 988 & 696 & 837 \\
173 & 341 & 680 & 845 & 1042 & 652 & 483 & 146 & 43 & 286 & . & 640 & 347 & 510 \\
510 & 640 & 411 & 285 & 376 & 17 & 147 & 457 & 560 & 808 & 523 & . & 248 & 244 \\
376 & 458 & 501 & 549 & 572 & 313 & 137 & 301 & 322 & 569 & 284 & 303 & . & 271 \\
358 & 413 & 232 & 341 & 573 & 215 & 209 & 319 & 448 & 685 & 417 & 199 & 222 & .
\end{bmatrix}$$

apm523-5

i) We built the symmetric matrix $\overline{D'}$ from asymmetric $\mathbf{D}'_{n\times n}$, of the form

$$\overline{D'} = \left[\begin{array}{c|c} K & D'^{T} \\ \hline D' & K \end{array}\right]_{2n\times 2n}$$

where, $K_{n\times n}$ consists of very large values from $\mathbf{D}'$. according to the algorithm [1].

Matlab code for the transformation: **burma14_ asym_ data.m**
Assymetric distance matrix: **INP.DAT**
Double sized symmetric matrix: **double_ size_ asym.txt**

ii) We used the Fortran code to solve the ASTP

apm523-5 Fortan codes: **main.f, newcdt.f**
Input data (a 14x14 assymetric matrix): **INP.DAT**
Output of the fortran exectutable: **TSP_ asym_ fortran_ results.txt**

iii) ATSP results for Burma14 with a 10% westerly wind

$$8 \to 14 \to 4 \to 5 \to 6 \to 12 \to 13 \to 2 \to 10 \to 1 \to 9 \to 7 \to 11 \to 3.$$

where the distance from city $i \to j$ corresponds to entry (i, j) in our asymmetric distance matrix. We wrote Matlab code to work through the vector representing the cities along our tour and add up the "total distance" of the trip. The "total distance" of the windy asymmetric tour is 4,704 miles.

Matlab code: **compute_ distance_ ATSP.m**.

c) Part C
The length found in part b) is not the acutal distance because we had made adjustments for the wind direction because that is the distance the plane would have felt as though it had travelled. In order to determine that actual distance along the ground that was travelled we needed to use the original distance $\mathbf{D}$ matrix before the 10% wind adjustments. If the leg we travelled was $i \to j$, then we will add the value of $\mathbf{D}(min(i,j), max(i,j))$.

For example: $8 \to 14$ has distance $\mathbf{D}^{'}(14, 8) = 319$ miles, while $\mathbf{D}(8, 14) = 355$ miles. After adjusting each leg of the tour we have a total distance travelled of 4,817 miles.

Matlab code: **compute_ distance_ actual.m**

Fun Question

In the Odyssey the westerly winds were given to Odysseus in a bag by Aeolus. This allows Odysseus to control his sail back home. Just as the ship is almost home, Odysseus's shipmates, suspecting that the bag was actually full of treasure, opened the bag and released the winds. These winds took the ship away from home and back to Aeolia in the east. Hence they were not able to get home as easily as they could have if they had left the bag alone.

References

[1] Kumar, Ratnesh & Li, Haomin. (2000). 'On Asymmetric TSP: Transformation to Symmetric TSP and Performance Bound'.