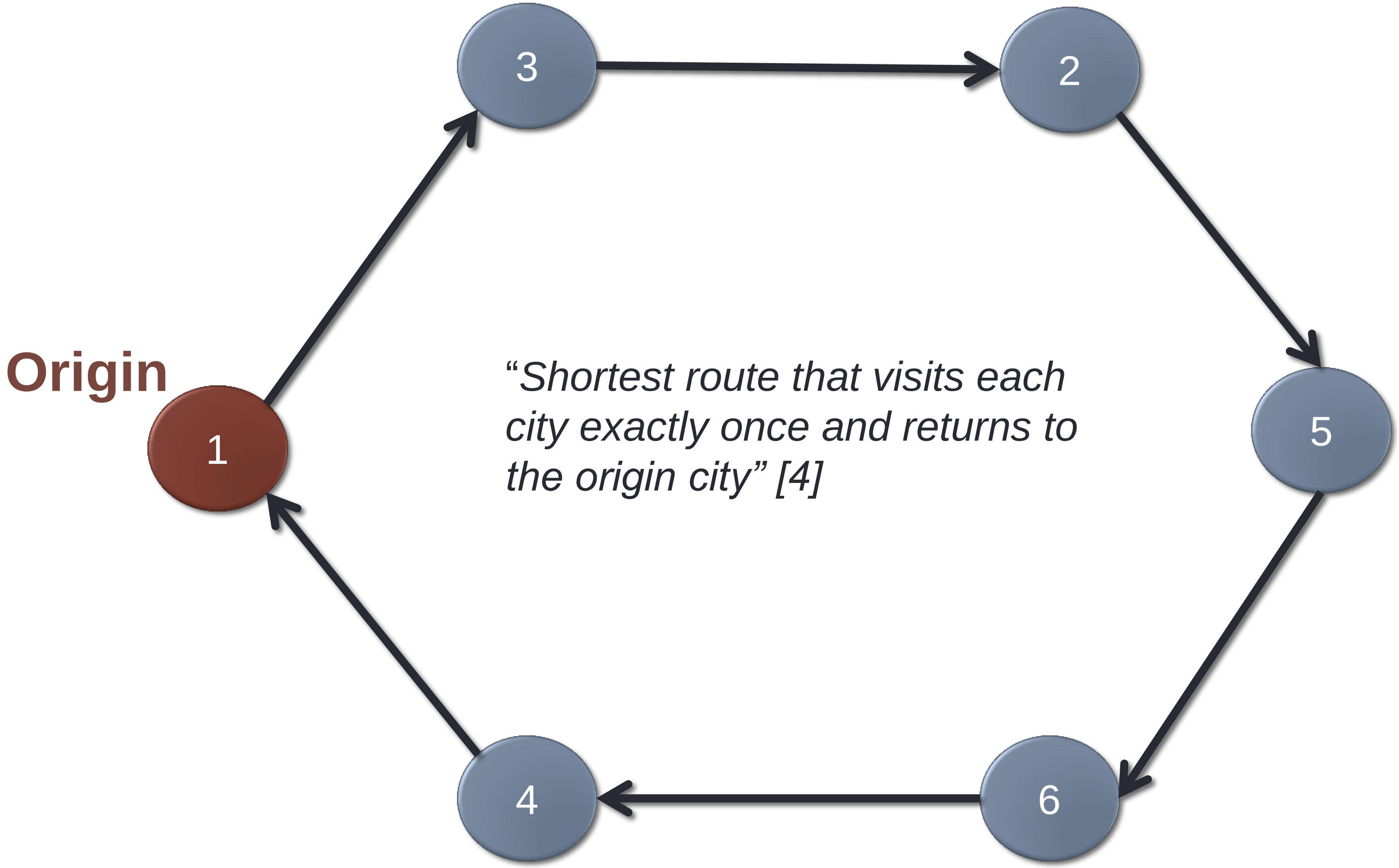
TRAVELLING SALESMAN PROBLEM

http://blog.codeeval.com/commute

Ramya Nair
CSCI - 5654
Final Project

General TSP



"Shortest route that visits each city exactly once and returns to the origin city" [4]

General TSP : Set-Up

- minimize Distance

so that

- **IN Criteria**
  - All cities must be arrived at from exactly one other city

- **OUT Criteria**
  - All cities must depart to exactly one other city

- **CONNECTIVITY Criteria**
  - There must be only one tour covering all cities

General TSP : Set-Up

minimize $\boldsymbol{c^T x}$

s.t.

- $\boldsymbol{A_{IN} \, . \, x \quad = 1,}$

  $where\ \boldsymbol{A_{IN}}\ matrix\ representing\ all\ the\ incoming\ edges\ for\ each\ vertex$

- $\boldsymbol{A_{OUT} \, . x \quad = 1,}$

  $where\ \boldsymbol{A_{OUT}}\ matrix\ representing\ all\ the\ outgoing\ edges\ for\ each\ vertex$

- $\boldsymbol{t_i \ - t_j \ + n . x_{ij} \ \le n - 1}$

  *Give a rank **t** to each vertex (except origin), ensure that **t** increases by one for every connected edge , restrict the upper bound of **t** to one less than the number of vertices*

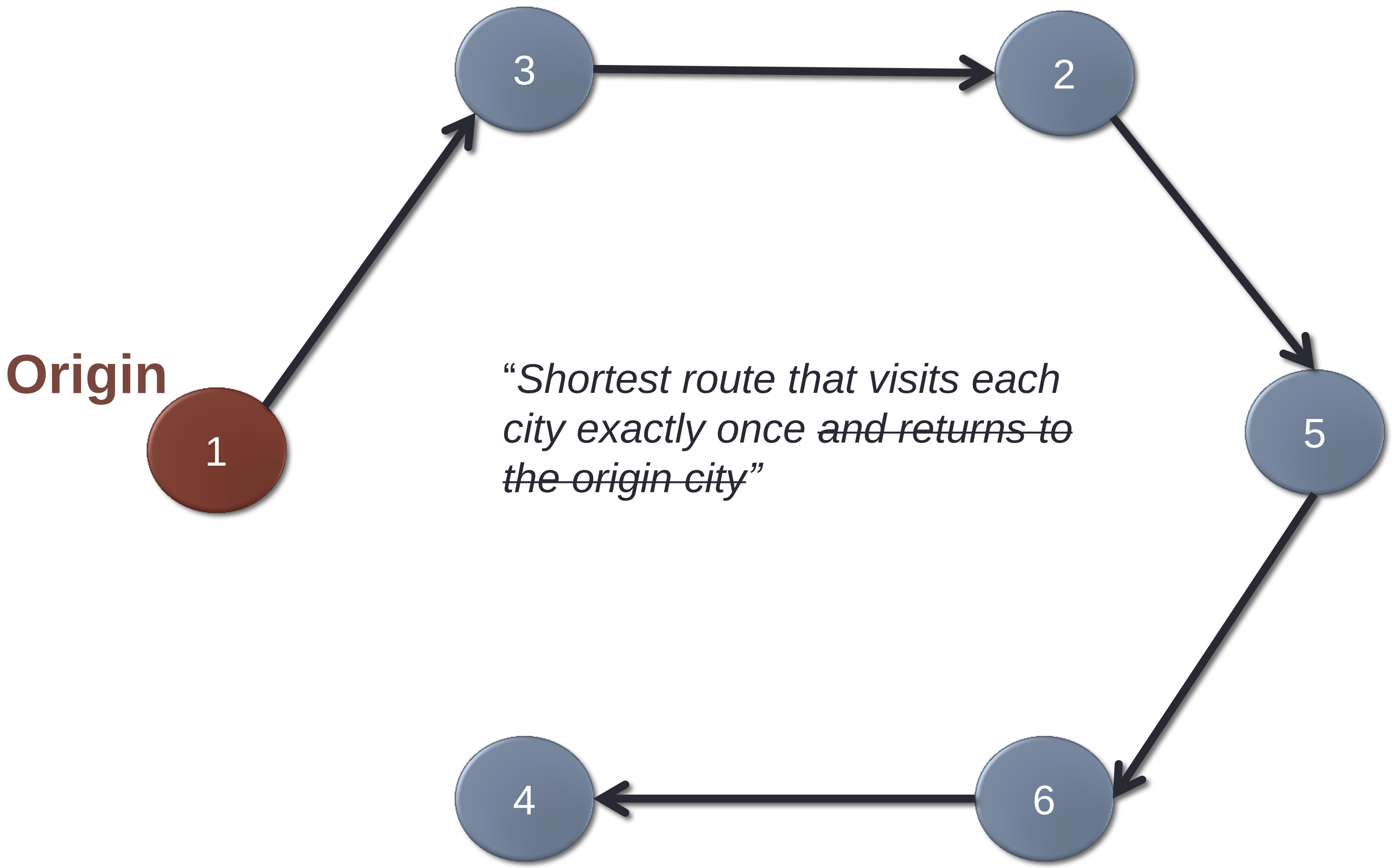
$n : number\ of\ vertices$

$A : matrix\ with\ each\ row = vertex, each\ column = edge$

$x : a\ vector\ representing\ all\ edges, 0 = not\ connected, 1 = connected$

$c : Distance\ vector\ for\ each\ edge\ (cost)$

<u>Coding challenge TSP</u>



“Shortest route that visits each city exactly once ~~and returns to the origin city~~”

Coding challenge TSP - setup

- minimize <u>Distance</u>

so that

- **IN Criteria**
  - All cities must be arrived at from exactly one other city **except the origin city**

- **OUT Criteria**
  - All cities must depart to exactly one other city **except the last city**

- **CONNECTIVITY Criteria**
  - There must be only one tour covering all cities **but it does not have to be a complete cycle**

Coding challenge TSP : Set-Up

minimize $\boldsymbol{c^T x}$

s.t.

- $\boldsymbol{A_{IN} \,.\, x \quad = 1},$
  $where\ \boldsymbol{A_{IN}}\ matrix\ representing\ all\ the\ incoming\ edges\ for\ each\ vertex\ \boldsymbol{except\ Origin}$

- $\boldsymbol{a_{OUT1} \,.\, x \quad = 1},$
  $where\ a_{OUT1}\ vector\ representing\ all\ the\ outgoing\ edges\ for\ origin\ vertex$

- $\boldsymbol{A_{OUT} \,.\, x \quad \leq 1},$
  $where\ \boldsymbol{A_{OUT}}\ matrix\ representing\ all\ the\ outgoing\ edges\ for\ each\ vertex\ \boldsymbol{except\ Origin}$

- $\boldsymbol{t_i \ - t_j \ + n.\, x_{ij} \ \leq n - 1}$
  Give a rank $\boldsymbol{t}$ *to each vertex ensure that* $\boldsymbol{t}$ *increases by one for every connected edge , restrict the upper bound of* $\boldsymbol{t}$ *to one less than the number of vertices*

$n\ :\ number\ of\ vertices$
$A\ :\ matrix\ with\ each\ row = vertex, each\ column = edge$
$x\ :\ a\ vector\ representing\ all\ edges, 0 = not\ connected, 1 = connected$
$c\ :\ Distance\ vector\ for\ each\ edge\ (cost)$

MATLAB - Implementation

- Steps

1. Read the text file parse for the url string
2. Obtain the xml file from google maps API
3. Parse xml file and form distance matrix
4. Set up the problem (as explained)
5. Implement branch and bound ILP (used *linprog* for LP relaxation)
6. Return the edge vector
7. The last n elements of the vector gives the route order

Branch and Bound

- Nothing special, mostly what was taught in class
- Pseudo Code:
  - Repeat the following until all loops exit
    - Get linprog solution to minimize the cost
    - If infeasible *return*
    - If unbounded *return*
    - If feasible solution
      - If integer solution update(global minimum Obj and global solution) and *return*;
      - If fractional solution ,
        - If current objective is greater than global minimum Obj , ***return*** (PRUNING)
        - Get (the first) edge with fractional value
        - Split the problem, update upper bound and lower bound for new 2 problems
        - For each of the two problems
          - If lower bound is greater than upper bound ***return*** (PRUNING 2)
          - Repeat this algorithm for updated bounds

Results

- From Codeeval

Output sample:

It must start from position 1

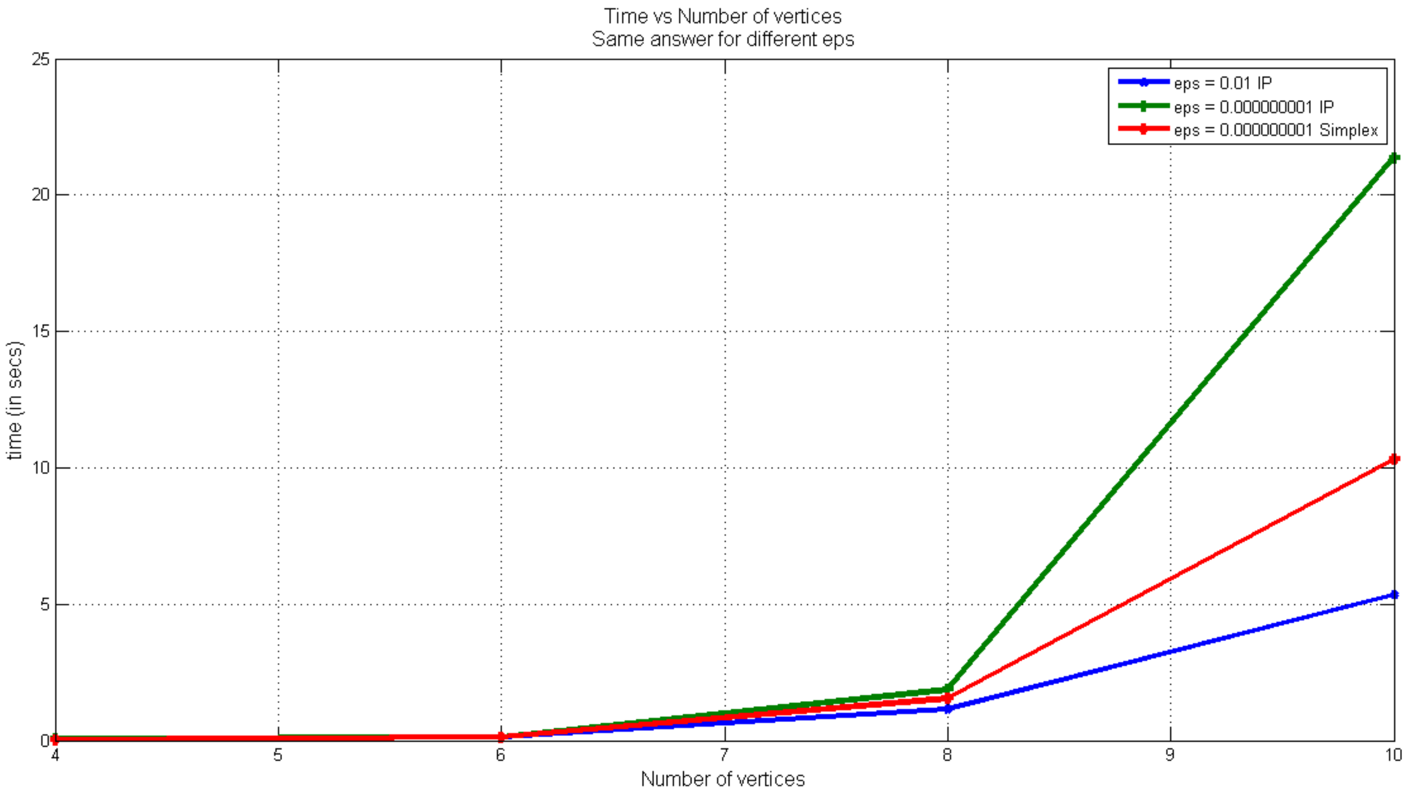
```
1
3
2
5
6
4
```

- My output

ans =

- 1
- 3
- 2
- 5
- 6
- 4

Scaling



Time vs Number of vertices
Same answer for different eps
eps = 0.01 IP
eps = 0.000000001 IP
eps = 0.000000001 Simplex
time (in secs)
Number of vertices
0
5
10
15
20
25
4
5
6
7
8
9
10

References

1. https://developers.google.com/maps/documentation/distancematrix/
2. http://www.cs.cmu.edu/afs/cs/project/pscico-guyb/realworld/www/slidesF08/linear3.pdf
3. http://support.sas.com/documentation/cdl/en/ormpug/63352/HTML/default/viewer.htm#ormpug_milpsolver_sect020.htm
4. http://en.wikipedia.org/wiki/Travelling_salesman_problem