M.Sc. in Mathematics (Finance)

2018/2019, 1º semester

Computational Methods

Project 1 – Solving the Traveling Salesperson Problem (TSP)

1. Introduction

The Traveling Salesperson Problem (TSP) is a well known problem that may be formulated as:

> Given a set of n cities connected by roads all with assigned distances, find the tour of minimum distance that a traveling salesperson should do to start and finish in an y city and visit all the other cities only once.

More formally, the problem is that of finding an Hamiltonian Tour of minimum cost in a graph, where the nodes of the graph (the cities) are connected by arcs labelled with costs (distances between cities).

graph_6_1.txt

```
6 15
1 2 84
1 3 70
1 4 5
1 5 1
1 6 98
2 3 74
2 4 96
2 5 51
2 6 35
3 4 65
3 5 95
3 6 70
4 5 35
4 6 98
5 6 19
```

2. Objective

Your goal is to get (approximate) solutions of the TSP, given a graph specified in a file such as that shown ("**`graph_6_1.txt`**") with the format as shown:

1. The first line indicates the number of nodes (**`6`**) and the number of arcs (**`15`**);
2. Each of the other lines specify an arc, i.e. the connected nodes and its weight (for example, nodes **`1`** and **`2`** are connected with an arc of weight **`84`**.

3. Implementation Notes

a. Use a function with the signature below

```
function [distance, Tour, exectime] = tsp(fname, mode)
```

where, for the graph represented in a file with name **`fname`**, returns the minimum **`distance`**, found in a **`Tour`**, represented as a vector with the order in which the nodes are visited. For example, for the graph shown a possible solution is **`T = [1,4,3,6,5,2]`** with distance **`d = 4+65+70+19+51+84 = 293`**.

b. Your function should read the file and produce an adjacency matrix that is symmetric (the distance between nodes p and q is the same between q and p).

c. Then you should fill the solution vector **`Tour`**, by assigning distinct nodes to consecutive elements of the vector. First you decide which node is the first (i.e. becomes **`T(1)`**, then chose the next node to become **`T(2)`** and so on, until all elements of **`T`** are chosen.

d. Notice that the elements of the vector must be all different. The distance should then be computed from vector **`T`**.

e. To choose the next node you may use several different heuristics (modes):

1) **`mode = [1]`**: Choose the node, among those not yet chosen, that is closer to the previous node;
2) **`mode = [2, nt]`**: Choose arbitrarily the node, among those not yet chosen. Repeat the search **`ni`** times and keep the best solution.
3) **`mode = [3, nt, ns]`**: Select up to **`ns`** nodes, among those that yet chosen, closer to the previous node, and chose arbitrarily between them. Repeat the search **`ni`** times and keep the best solution.
4) **`mode = [3, nt, ns]`**: Select up to **`ns`** nodes, among those that yet chosen, closer to the previous node, and choose between them, with a probability that is proportional to the inverse of the distance (the farthest the node, the least probable it is to be chosen).

f. To return the execution time, **`exectime`**, use the predefind function **`cputime()`**.

4. Final Report

Write a small report explaining how you carried out your simulation, namely the functions and data structures that you used. Moreover, report (best tours and their distances, together with the execution times) on the solutions obtained in graphs in file **`tsp_graphs.zip`**.

The report, as well as the files with your code, must be sent by email to the lecturer (**pb@fct.unl.pt**) with subject **Project_MC_1_by_XXXXX+YYYYY** (where XXXXX and YYYYY are the numbers of the students - max 2 per group), **no later than** Friday, 21 December at 23:59.