Traveling Couple

Advanced Algorithms 2013, Project 2

Noy Rotbart

14.5.13

This is the second mandatory assignment for the course Advanced Algorithms 2013. The Assignment should be solved in groups of 2 or 3 students. Permission for individual assignments is only granted in special cases. The assignment is due Fri. 24th of May at 22:00. To pass the assignment the group must have completed most of the questions satisfactory. To be allowed to resubmit the group must have made a reasonable attempt at answering most of the questions. The resubmission is due Mon. 2nd of June at 22:00. Completed assignments must be uploaded to Absalon (this page). All descriptions and arguments should be kept concise while still containing relevant points. The assignment has two parts, where the second is mainly about implementation.

Software and programming language

There are no requirements on your choice of programming language, but you should select one which has bindings for a Linear Programming solver (LP-solver). The problems you are asked to solve are relatively small, which means that the ease of use of the LP-solver is much more important than its efficiency. We recommend using either the pulp-or library for python, the Matlab default linear solver, or lp_solve for Java.

1 Exercise 1: Theory

Rasmus and Maj wish to see all the monuments in a city. To see monument m they must visit it or visit another monument at most d meters away from

it. In both cases we say that monument m is visited.

Let $G = (V, E)$ be a complete undirected weighted graph where $v_1 \dots v_n$ are the monuments and the weights d_{ij} $(i, j \in \{1 \dots n\})$ are the distances between monuments. The *Traveling Couple Problem*, or *TCP*, is the optimization problem of finding the shortest path where all monuments are visited.

Exercise 1.1

- Give a formal definition of the problem. In addition, write a related decision problem and the corresponding language.
- Prove by reduction to *HAM-CYCLE* that the decision problem you formulated is NP-complete. What does that say about the time complexity of the original problem?

Exercise 1.2

Last year, a friend of Rasmus and Maj visited the same city. He tried to solve the TCP using a metaheuristic but the solution was sub-optimal and he got tired of walking. Rasmus, therefore wish to solve TCP to optimality using a Branch and Bound (BnB) algorithm. The hard part about BnB is to find a good lower bound. Rasmus, naive as he is, suggests using the size of a one-tree as a lower bound on the optimal TCP tour.

Give an example of a set of points in the plane and a d-value where even the smallest one-tree is larger than the optimal TCP tour.

Exercise 1.3

Rasmus and Maj realize that getting good lower bounds for the TCP is tricky so they decide to use a linear program to get a lower bound. To do this they must first write TCP as an integer linear program (ILP).

- Write the TCP as an ILP.

  Hint 1 : Let $v(i) = \{j \in J | d_{ij} \leq d\}$ denote the set of vertices in the vicinity of i .

  Hint 2: In order to eliminate sub-tours, you may introduce constraints on both the vertices and the edges.

- Which constraints can Rasmus and Maj remove from the ILP formulation of TCP so that the optimal value becomes a lower bound of TCP?

Exercise 2: Practice

Attached is a Java-implementation of a BnB algorithm without the lower-bound method. There is a description of the method in the file.

Exercise 2.1

Implement the lower-bound method using some variant of the ILP for TCP. Describe which constraints you decided to remove or relax and why. Your implementation should be attached when handing in the assignment with a description thereof.

Exercise 2.2

Report the optimal TCP tour lengths of Problems 1, 2 and 3 (from the file Graphs.java) as well as the number of BnB-nodes that were evaluated.

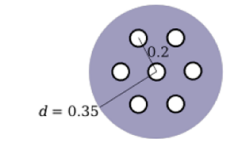

Figure 1: graph1.java

Figure 2: graph2.java

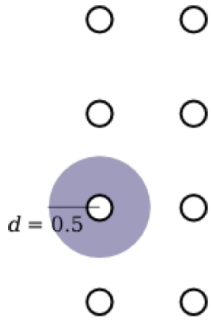

Figure 3: graph3.java

Exercise 2.3

This Ex. is optional.

We offer a trophy for the first group that can give an optimal solution to the fourth problem using any method of their choice (FlorenceGraph.java). On a side note, the previous best heuristic solution was 22.54km.

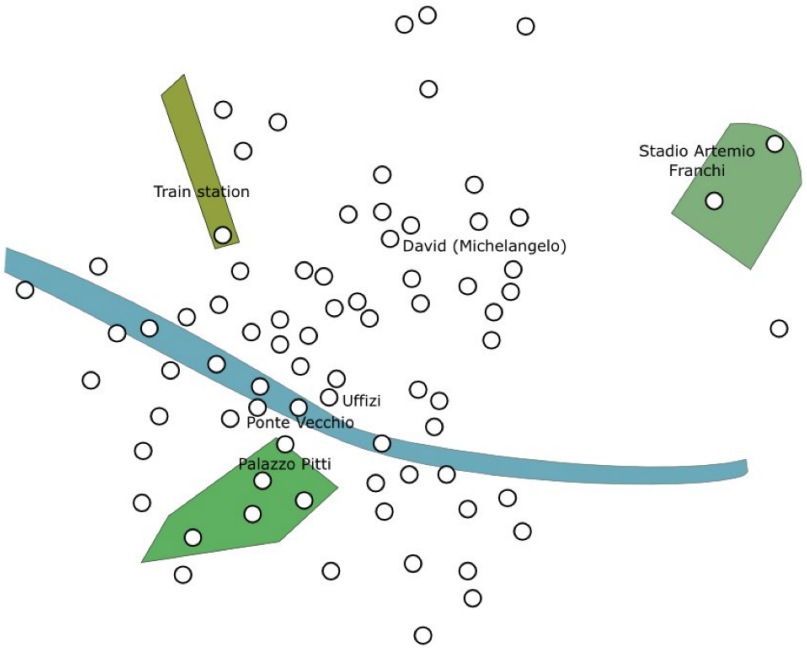

Figure 4: Graph representing monuments in Florence. Edges are implicit. The graph is complete and the weight of an edge (u, v) is the planar distance from u to v. The viewing distance is set to $d = 200$ meters. FlorenceGraph.java