# Project 2: Branch and Bound

*Advanced Algorithms 2011*

## INTRODUCTION

This is the second mandatory assignment for the course *Advanced Algorithms* 2011. Assignments must be completed in groups of 2 or 3 students. Permission for individual assignments is only granted in special cases. The assignment is handed out on Tuesday, the 10/05/11 at 12:00 and is **due on Wednesday, the 18/05/11 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 solving most of the questions. The resubmission is due Wednesday, the 1/6/11 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 pulp-or library for python, the default Matlab linear solver, or lp_solve for Java.

## PART 1

A couple of tourists wishes to see all monuments, $i = 1...n$, in a city. To do this they must either visit the monument or visit another monument at most *d* meters away from it. They therefore create a fully connected undirected weighted graph where the monuments are the vertices and the weights, $d_{ij}$, are the distances between monuments. The couple wishes to find a shortest cycle that enables them to see all monuments and get back to the original position. They call this problem the Traveling Couple Problem (TCP).

**Question 1.1**

Part A : Give a formal definition for the problem. In addition, write a related decision problem and the corresponding language.

Part B : Prove by reduction from HAM-CYCLE decision version of the TCP problem is NP-complete.

Formally, Prove : $HAM-CYCLE \leq_p TCP$ . What does that say about the time complexity of the original problem?

### Question 1.2

Last year, the friends of the two tourists visited the same city. They tried to solve the TCP using a metaheuristic but the solution was sub-optimal and they got tired from walking. This year's tourists 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. The guy, naive as he is, suggests using the size of a 1-tree as a lower bound on the optimal TCP cycle.

Give an example of a set of points in the plane and a d-value where even the smallest 1-tree is larger than the optimal TCP cycle.

### Question 1.3

The tourists 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: let $\nu(i) = \{j \in V \,|\, d_{ij} \leq d\}$ denote the set of vertices in the vicinity of *i*).

### Question 1.4

Which constraints can they remove from the ILP formulation of TCP so that the optimal value becomes a lower bound of TCP.

## PART 2

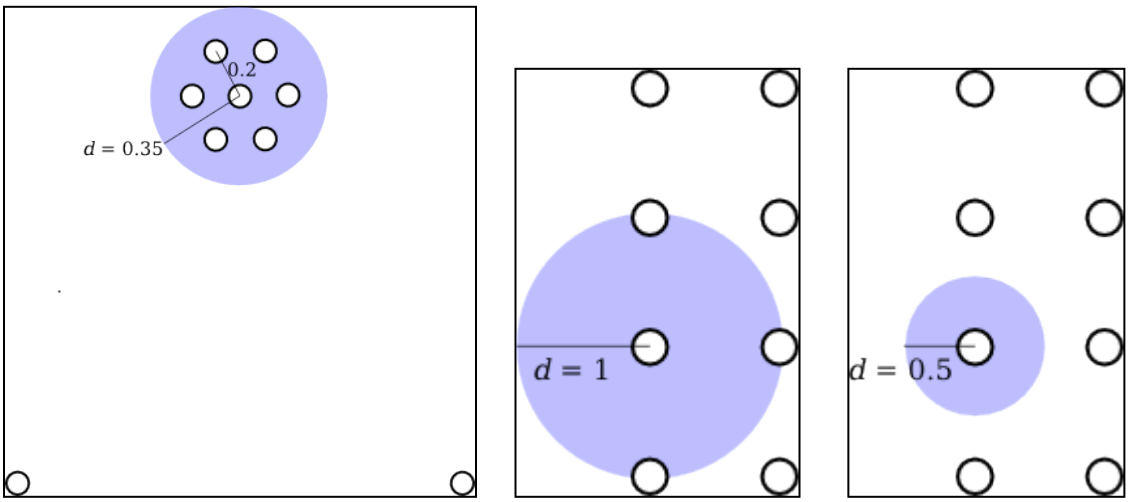
Attached is a Java-implementation of a BnB algorithm without the lower-bound method. There is a description of the method in the file.

### Question 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.

### Question 2.2

Report the optimal TCP cycle lengths of Problems 1, 2 and 3 (Graphs*.java) as well as the number of BnB-nodes that were evaluated.If you intend to use the CSV files please notice that the difference between the problems 2 and 3 is just in d. Therefore, you should use the same file Graph2.csv , with the required modification.



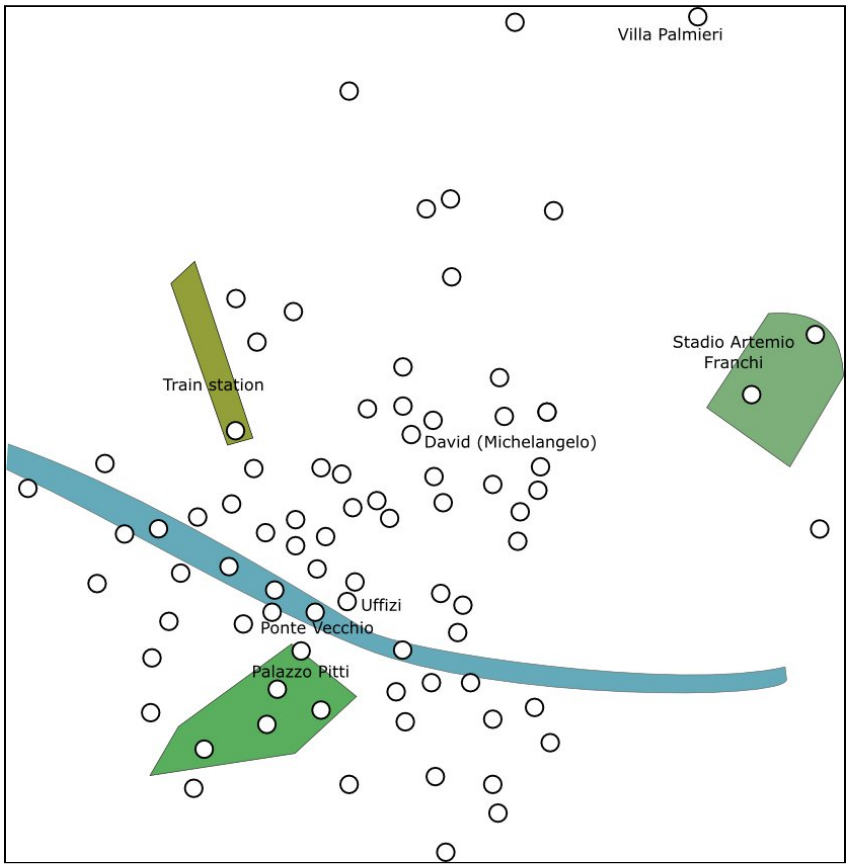

Graph1.java Graph2.java Graph3.java

**Figure 1 : The graphs**

**Question 2.3 (optional)**

A prize will be given 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, last year's best heuristic solution was 22.54km.




**Problem 4**: Graph representing monuments in Florence. Edges are implicit. The graph is fully connected, 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**