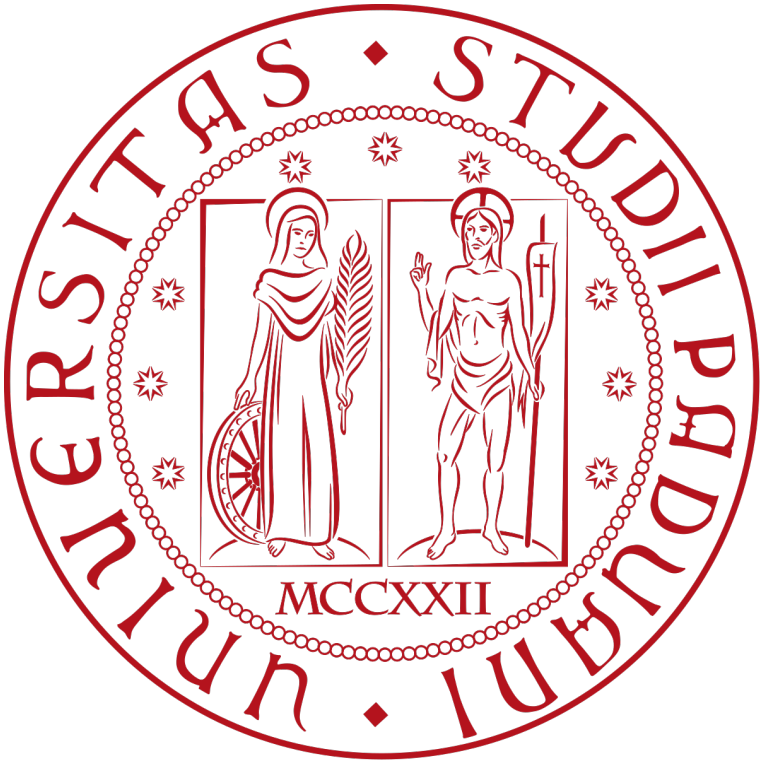
Progress Report

Operational Reseach 2

Thomas Porro

August 2021



School of Computer Engineering

Contents

1 The Problem

In this report we are going to describe, analyze and implement solutions for the Travelling Salesman Problem (from now on it will be called TSP).
Essentially the problem have this type of formulation is the following: "Given a list of cities and the distances between eachother, find the shortest path that connect all the cities". This could be "translated" to find the shortest (or the one with the lowest cost) hamiltonian circuit given an oriented graph $G = (V, A)$, where V are the cities of the problem and A are the paths that connect each city to the other ones.

Matematically the problem is the following. We start numbering all the cities that we have, from now on they will be called nodes, then we introduce a decisional variable x_{ij}

$$x_{ij} = \begin{cases} 1 & \text{if the arc } (i,j) \in A \text{ is chosen in the optimal solution} \\ 0 & \text{otherwise} \end{cases} \tag{1.1}$$

Now we can describe the first formulation of the problem:

$$min \sum_{(i,j)\in A} c_{ij} x_{ij} \tag{1.2a}$$

$$\sum_{(i,j)\delta^-(j)} x_{ij} = 1, \quad j \in V \tag{1.2b}$$

$$\sum_{(i,j)\delta^+(j)} x_{ij} = 1, \quad i \in V \tag{1.2c}$$

$$x_{ij} \geq 0 \text{ intero}, \quad (i,j) \in A \tag{1.2e}$$

In these equations we use the value c_{ij} as the cost of the path from the node i to the node j. The equations 1.2b and 1.2c lead to the fact that each node must have only one arc incoming and one arc outgoing.

2 Code setup

In order to implement the models to be solved we decided to use the common and powerful tool IBM CPLEX. Usually this software isn't free but due the academic usage it was made available for all the students that needed it.
CPLEX allow its user to decide which programming language to use between Python and C; in this project we used C.

To visualize the nodes and the paths found by our program we used a Gnuplot which is a command-line driven utility. Its code is protected byt copyright but the download is completely free. The sofware needs to be installed on the machine where the code is executed because Gnuplot is executed as a pipe: in particular before the plotting all the data is wrote to a file (according to the documentation) and than Gnuplot read and create the plot from that file.

To build the performance profiles in this report we used a python program written by D. Salvagnin (2019).

The first thing we did was build a parser capable of interpreting the TSP problems provided by the TSPLIB. The main data to save was the number of nodes, the coordinates of the nodes (they will be or relative coordinates that describe the position of the nodes or real world coordinates), the type of distance function to use (for example when are used real world coordinates the distance function need to consider the sphericity of the world). For each tsp problem we assume that the datafile contains a complete graph, so each node is directly connected with all the others nodes.

In my particular case all the project was developed on a linux machine with Ubuntu 20.04.

2.1 CPLEX environment

In order to work properly CPLEX needs to build his internal data structure to hold all the information needed to solve the problem. So the first thing to do is to create a pointer to the environment of Cplex through the CPXopenCPLEX(&error): this function will return a pointer to the CPLEX environment that will be needed to use his entire library.
Once the environment is build CPLEX needs an additional data structure to hold the constraints of the optimization

problem we want to solve, in order to use it we build an empty object using the funtcion PXcreateprob(env, &error, "TSP"): this will return a pointer to the problem where we will write all the constraints that we need.

2.1.1 Variable creation