Paris Metro
Project Report

METAHEURISTICS UNIT – DSTI

SUBMITTED BY: MAHER SEBAI (A18)

27 JANUARY 2019

Executive Summary:

- Paris Metro Network Abstraction: positive weighted directed graph, Time as weight
- Problem Formulation: finding shortest Hamiltonian Path given a time budget
- Solution Infrastructure:
  - Rstudio : development IDE
  - igraph package: Paris Metro network graph modelisation and distance matrix computation
  - TSP package: Traveling Salesman Problem Solver
- Results:
  - 2 Network Models used: full Network and Compact Network
  - 9 classical TSP Algorithm applied and compared: each run 10 times + two-opt enhancement activated
  - Full Network best result: **07H13m03s** using **Nearest Neighbour Algorithm**
  - Compact Network best result: **05H28m47s** using **Farthest Insertion Algorithm**

Solution Detailed Description

- Problem Formulation and adopted strategy
- Data Cleaning
- Solution Implementation in R
- Results Charts
- Future enhancement

Problem Formulation

- Goal: visit maximum of Paris Intra-muros Stations with time budget of 20 Hours
- Provided data: data file representing Paris Metro Stations as Vertices, and trip duration from station to next station
- The problem can be formulated as finding the shortest Hamiltonian Path in a graph.
  - An Hamiltonian Path in a Graph is the one that links ALL the vertices of the graph ONLY ONCE
  - If the path is a cycle (returns to the departure vertices) then we have the classic traveling salesman problem formulation
- The Graph being a model with station as vertices and edges being the trip duration
  - The graph is weighted and directed
- Two Network Models Proposed:
  - Full Model: Hub Station represented with several Vertices
  - Compact Model: Hub Station represented with a Single Vertices

Data Cleaning

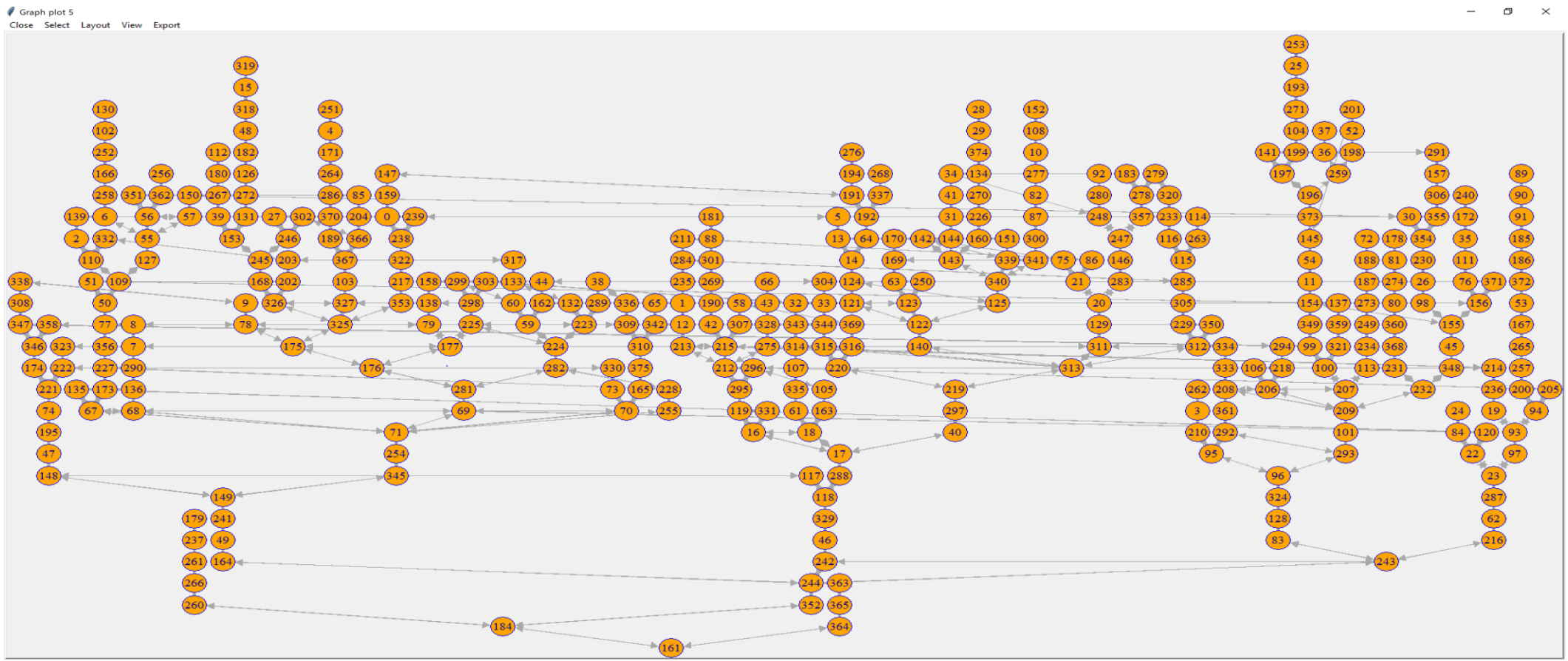
- Data file should be converted to “data frame” in Rstudio: a reliable column separator is needed
- Data file split in two files: Metro_vertices.txt + Metro_edges.txt
- Stations that are not in Paris intra-Muros are manually removed (vertices and edges)
- Metro_vertices.txt processed to change the column separator from “ ” to “;” to avoid word spaces in Station names. Regular expression used:
  - sed -i -e 's/(\d+)\s(.+\b)/\1;\2/g' Metro_vertices.txt
- The construct the Compact Network, the Hub station keep only their minimum vertice ID as Unique ID. R-code Snippet:

```
metro_stations$Unique_ID <- sapply(metro_stations$Station, function(x) { min(which(metro_stations$Station == x)) - 1})
metro_edges_compact$From <- sapply(metro_edges$From, function(x) {metro_stations$Unique_ID[which(metro_stations$ID == x)]})
metro_edges_compact$To <- sapply(metro_edges$To, function(x) {metro_stations$Unique_ID[which(metro_stations$ID == x)]})
```

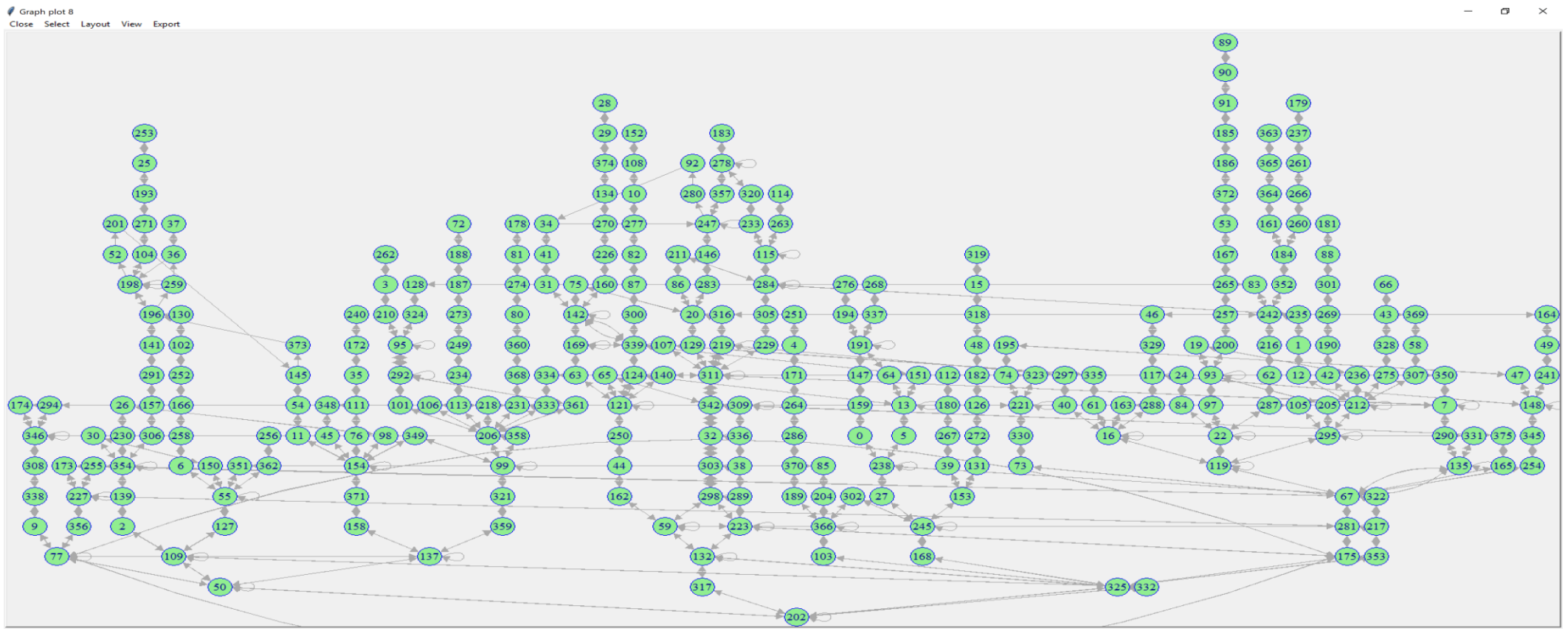
Solution Implementation in R

- igraph and TSP packages library loaded
- vertices and edges files loaded in 2 R “data frames”
- 2 weighted directed igraph class instances is constructed thanks to these data frames.
  - first igraph instance for full Network
  - Second igraph instance for compact Network
- The adjacency Matrix for these instances are sparse Matrix which is expexted as the graph is not fully connected (i.e not all the stations have direct connection to all other stations)
- a distance Matrix is constructed: internally a short path algorithm is called for all pairs of vertices (Dijkstra Algorithm) to construct a virtually fully connected graph reusing the existing connections.
- NOTE: the returned distance Matrix is not Symetric, Because the graph is directed and some trips are not bidirectional

Metro Full Network Graph Model



Metro Compact Network Graph Model



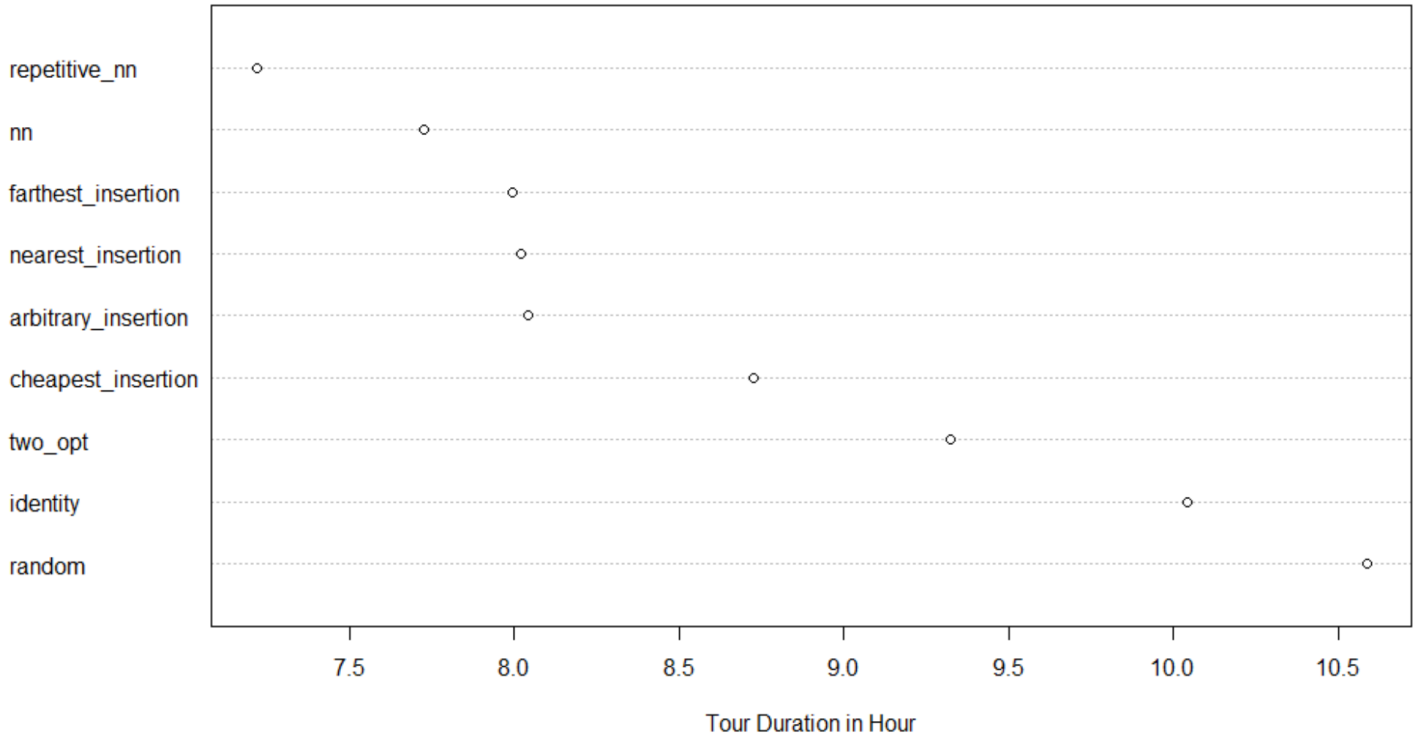
Solution Implementation in R

- The asymmetric distance Matrix allow as a construct an Asymmetric TSP class. It can be reformulated as a Symmetric TSP class but this is not needed as the solver can handle this type of class.
- The trick to Solve the Shorted Hamiltonian Path (acyclic solution) using TSP (cyclic solution) Solver is to insert a “Dummy Station” that has zero time/distance from all other stations. This “dummy station” will be the departure and arrival vertices for the TSP “cyclic” solution

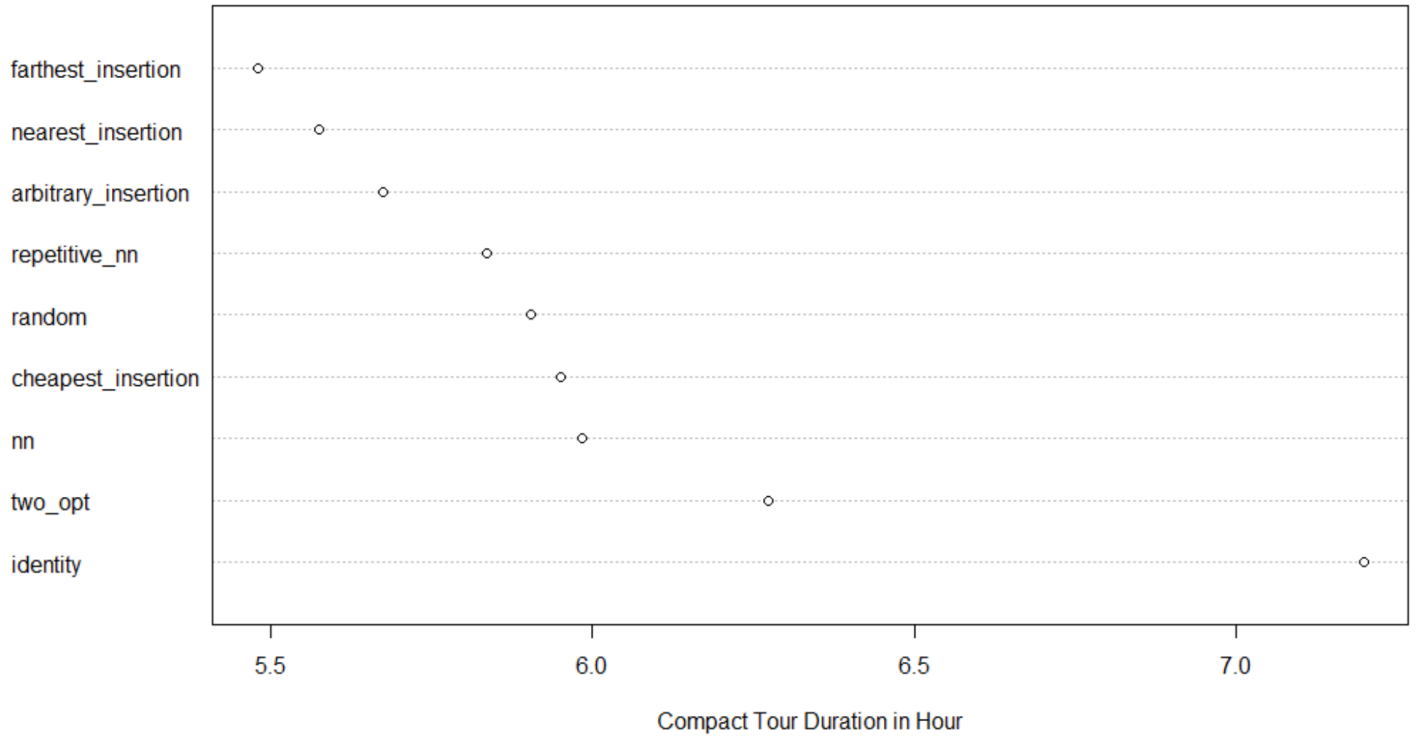
Solution Implementation in R

- TSP Solver called with Algorithm argument: List of Algorithm used:
  - "identity", "random" return a tour representing the order in the data (identity order) or a random order.
  - Nearest, farthest, cheapest and arbitrary insertion algorithms for a symmetric and asymmetric TSPs
    Nearest neighbor and repetitive nearest neighbor algorithms for symmetric and asymmetric TSPs
  - Two edge exchange improvement procedure
- The solver return a Tour object with tour length and vertice route path description
  - Full Network best result: **07H13m03s** using **Nearest Neighbour Algorithm**
  - Compact Network best result: **05H28m47s** using **Farthest Insertion Algorithm**

Full Network Result chart



Compact Network Result chart



R-Code file:

- Please refer to R-code for full solution implementation: “metro_project_maher_sebai.R”

Futur Enhancements

- Tour solution visualization in graph
- Tour visualization is Paris Metro Map
- Test other Non-TSP algorithms