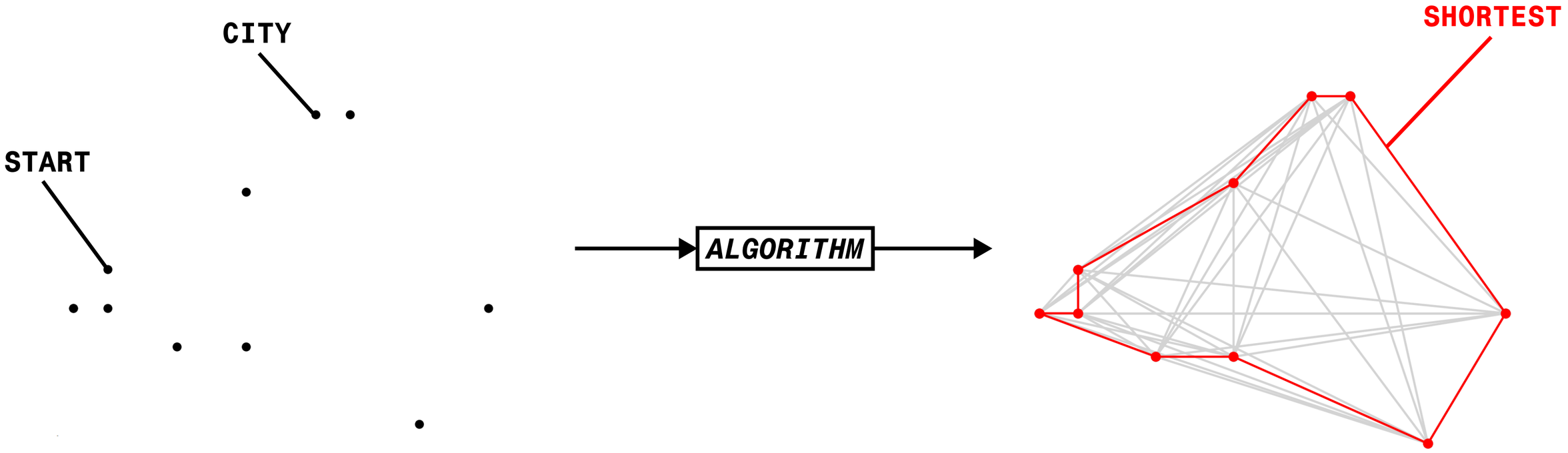
TRAVELING SALESMAN PROBLEM

RECURSIVE DIVIDE AND CONQUER

WHAT IS THE TSP?

The Traveling Salesman Problem (TSP) is a classic problem in the field of computer science and operations research, especially in the areas of optimization and algorithms. The problem is deceptively simple to describe but notoriously difficult to solve as the scale of the problem increases:

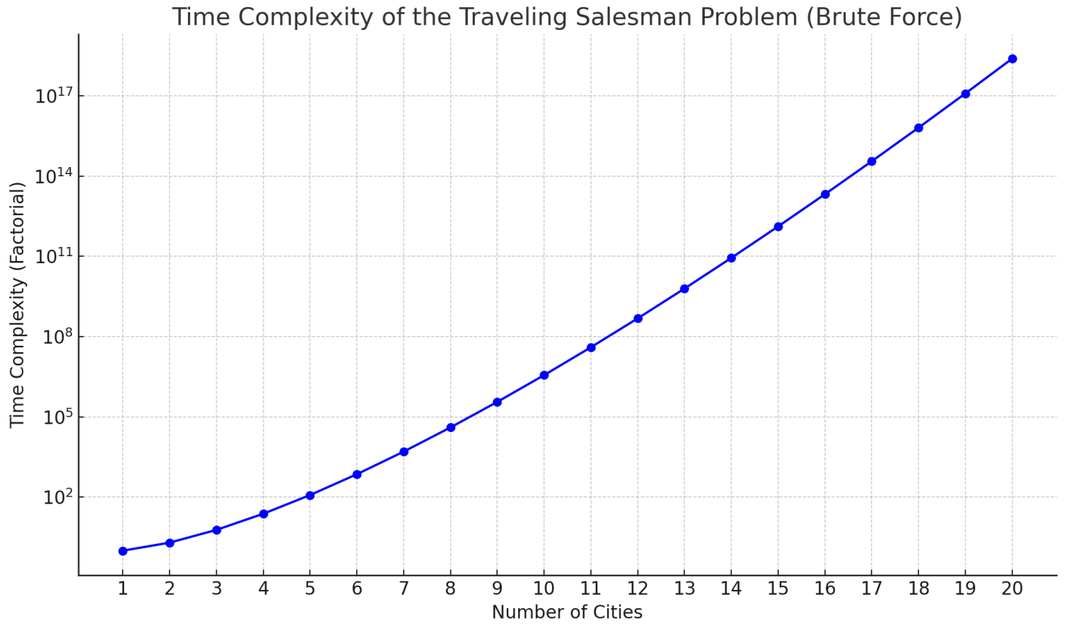
"Given a list of cities and the distances between each pair of cities, the task is to find the shortest possible route that visits each city exactly once and returns to the origin city."



TIME COMPLEXITY

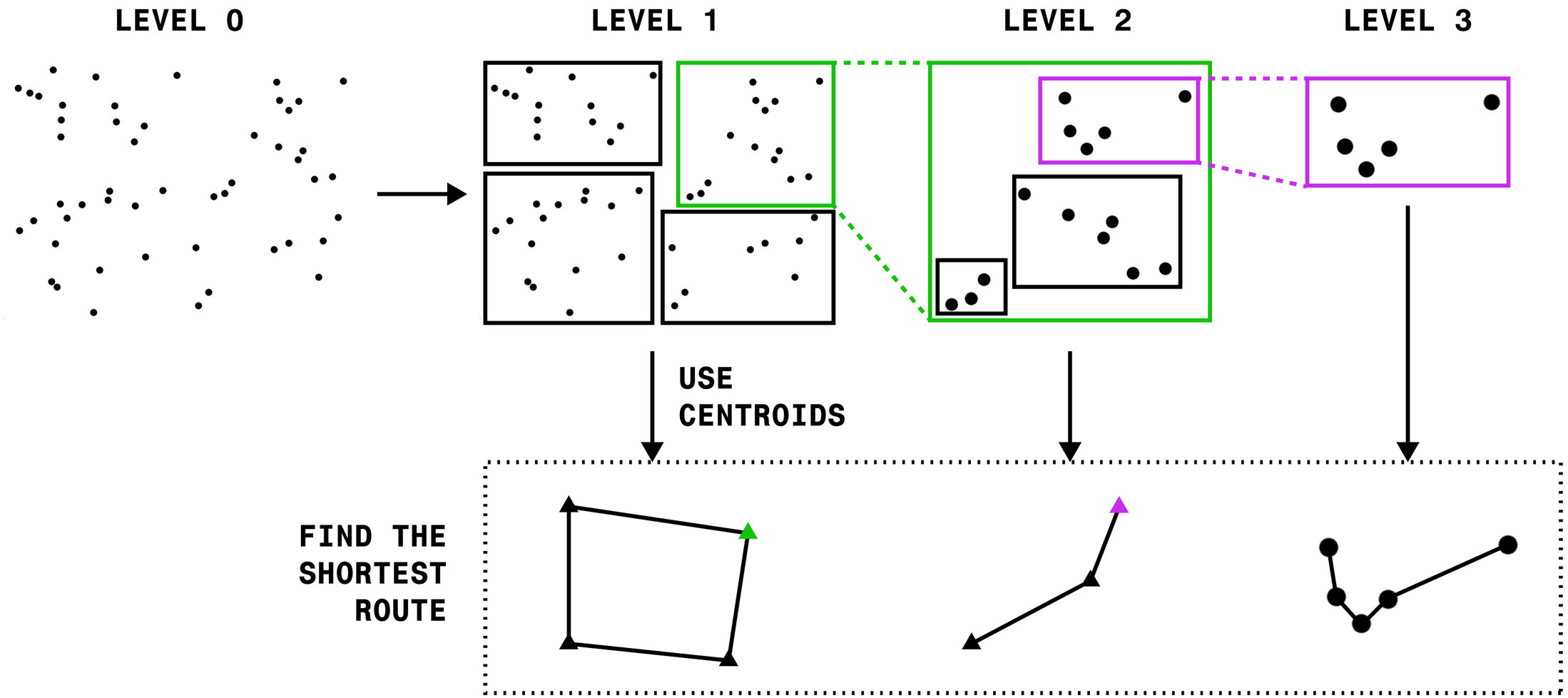
Time complexity is a concept used to quantify the amount of time an algorithm takes to complete as a function of the length of the input data. It provides a way to estimate the efficiency of an algorithm and predict how it scales with increasing data sizes.

For TSP, the time complexity of examining every possible route (to determine the shortest one) increases factorially with the number of cities (n), noted as O(n!). This means the time required grows much faster than the input size, leading to a situation where even with powerful computers, solving TSP for many cities becomes infeasible due to the enormous amount of time and computational resources required.



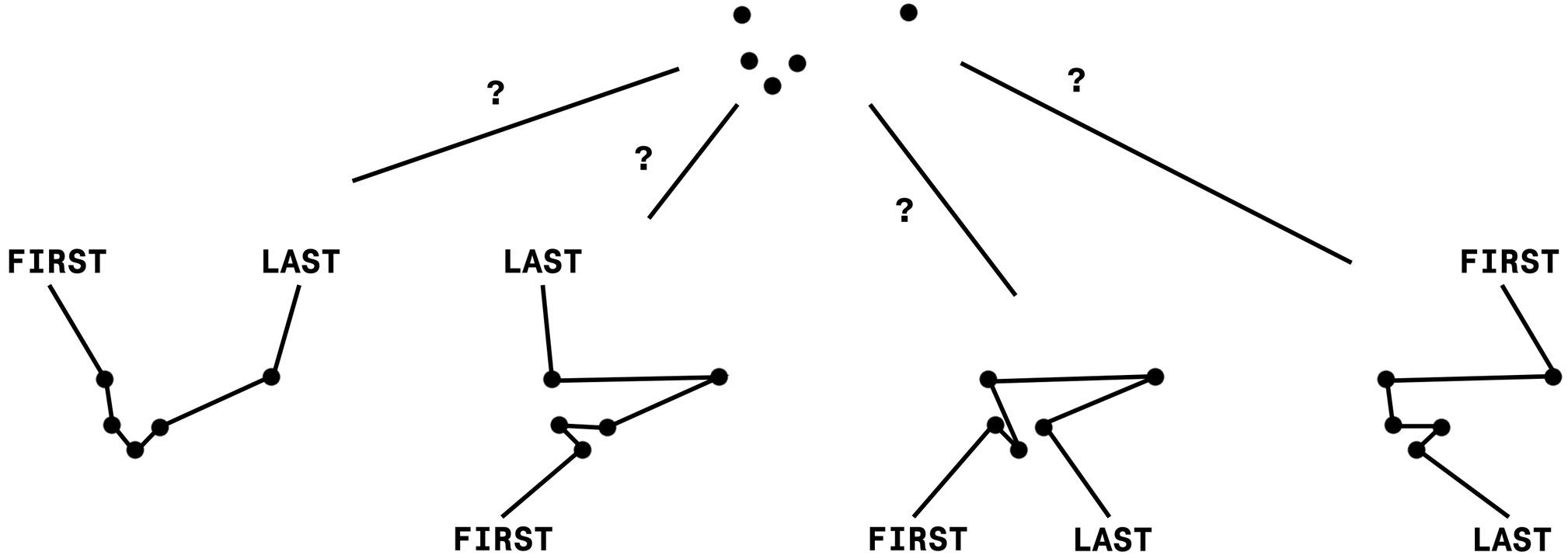
RECURSIVE DIVIDE AND CONQUER ALGORITHM

Divide - Simplify - Solve - Repeat



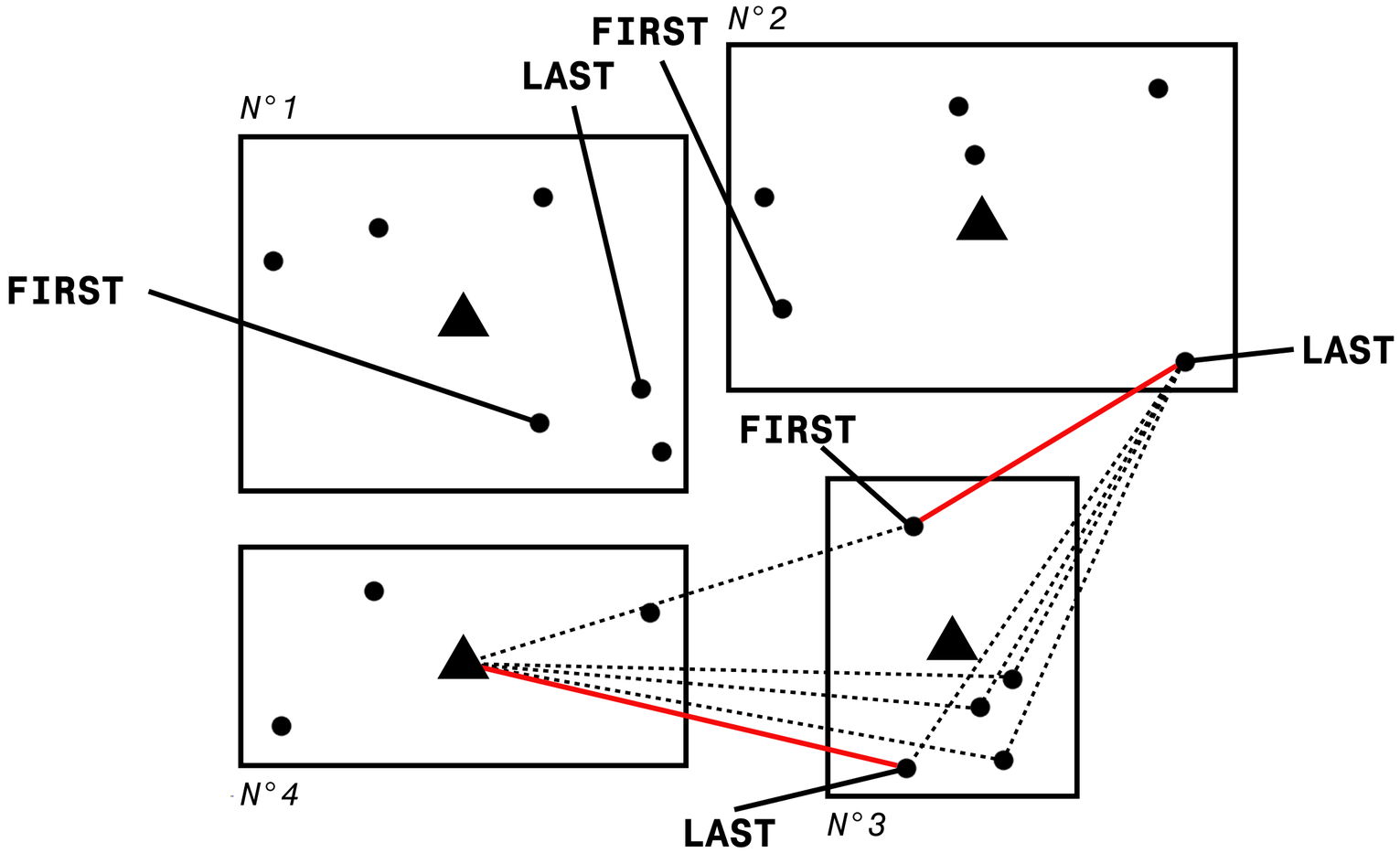
PARAMETERS

-- Number of clusters (groups) generated at each level
-- The maximum number of cities in a cluster required to solve to get the shortest path (=Threshold)
-- Clustering algorithm
-- Algorithm used to get the shortest path
-- Algorithm to get the first and last point (cluster or city) in each cluster:



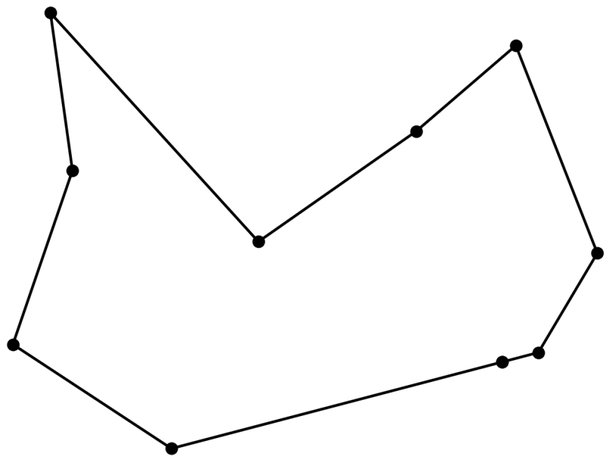
MY CONFIGURATION

-- Number of clusters (groups) generated at each level: NOT FIXED
-- Threshold: NOT FIXED
-- Clustering algorithm: K-means
-- Algorithm used to get the shortest path: Brute force
-- Algorithm to get the first and last point (cluster or city) in each cluster: FIRST=the closest point to the last point of the previous cluster / LAST=the closest point to the centroid of the next cluster:

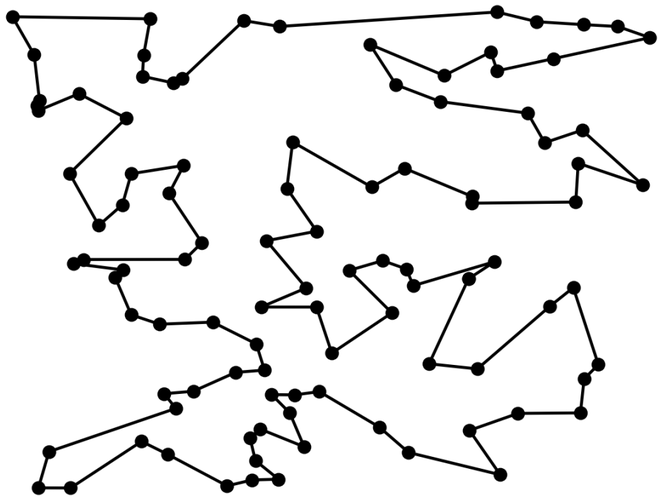


RESULTS

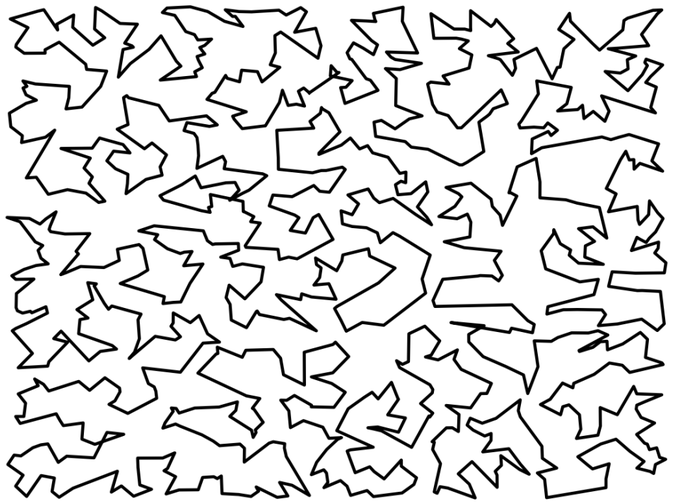
Number of clusters: 7 / Threshold: 8



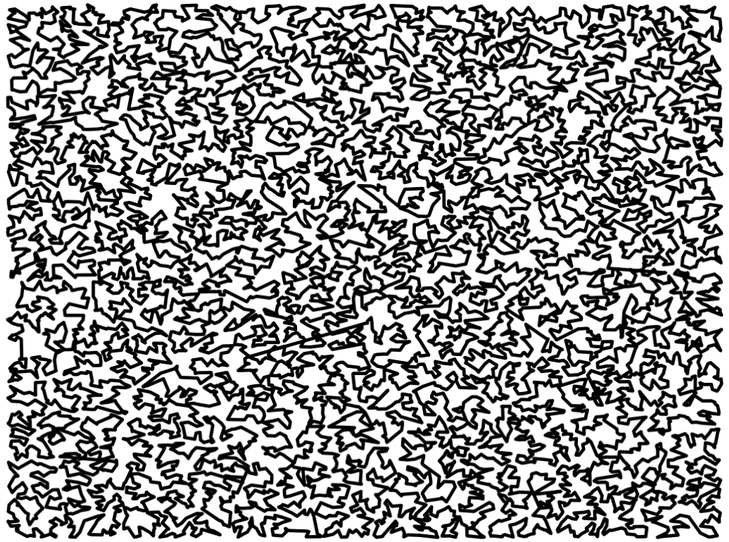
10 cities - 0.23s



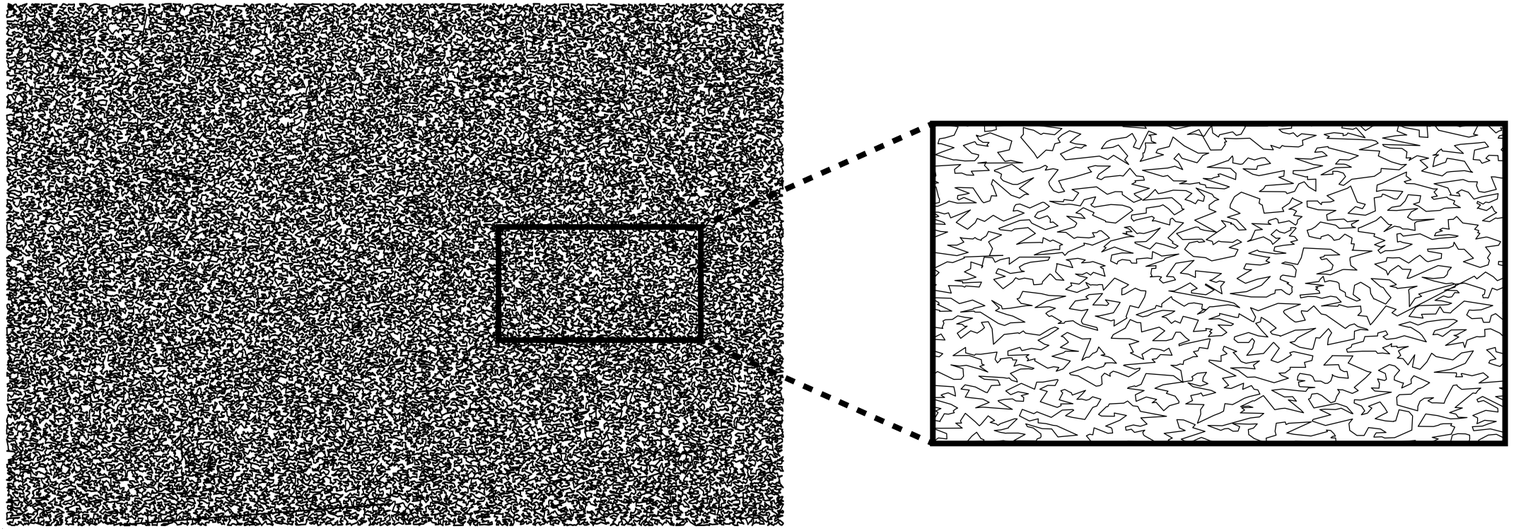
100 cities - 0.66s



1000 cities - 5.79s



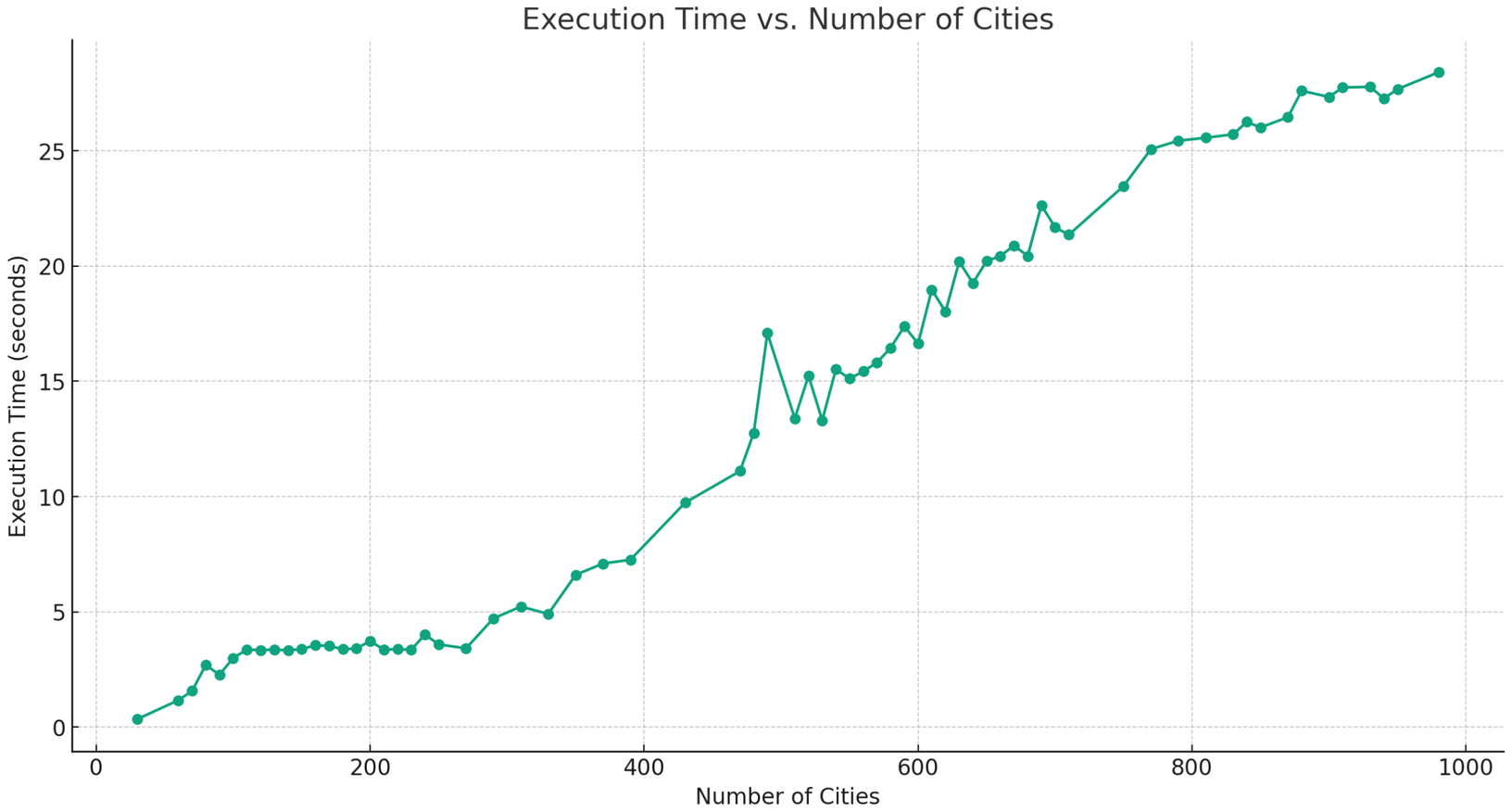
10,000 cities - 61.36s



100,000 cities - 843.85s

RESULTS

Number of clusters: 8 / Threshold: 9



Execution time = cities generation + solve TSP + data save

PERFORMANCE

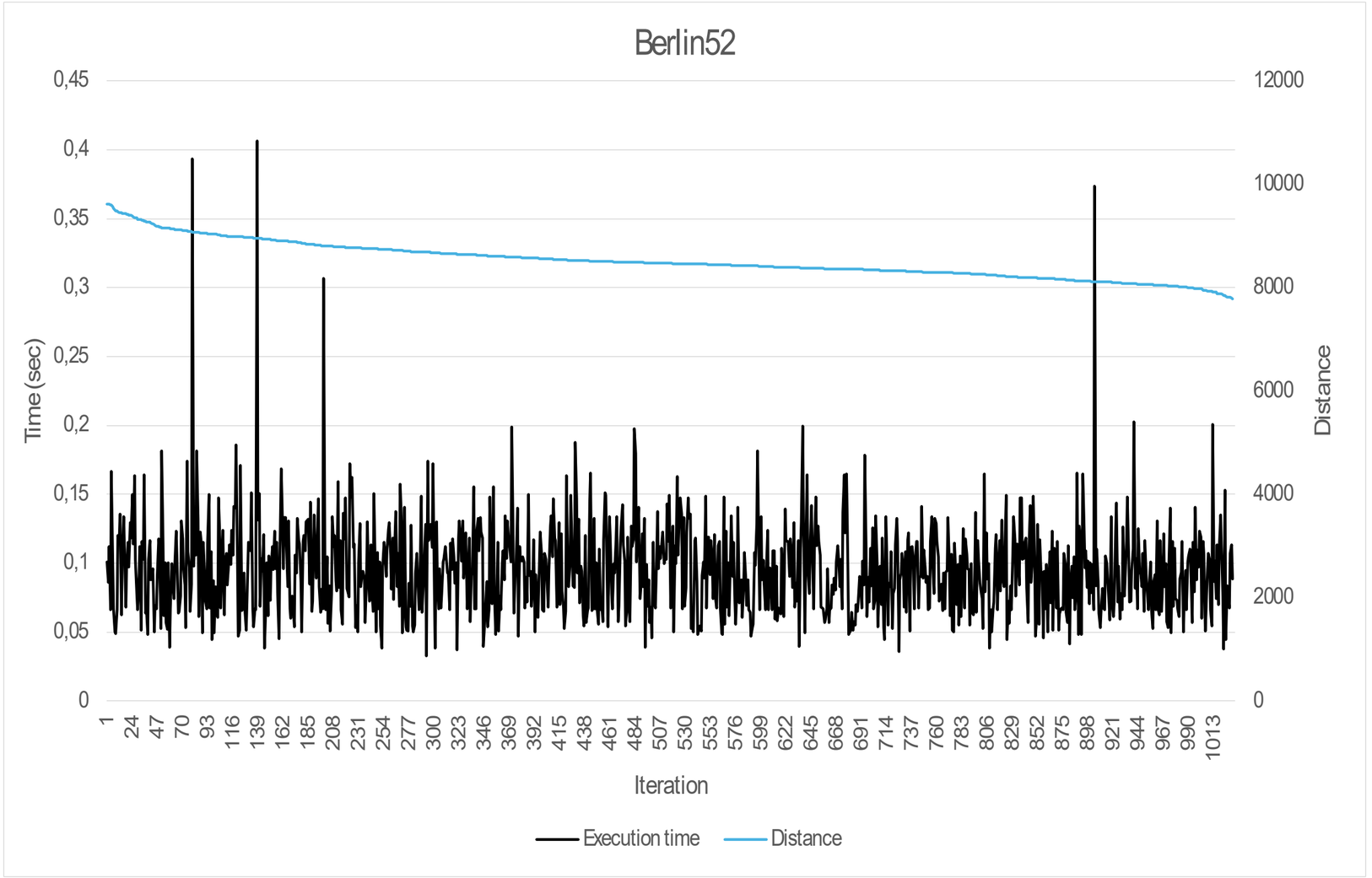
Performance on Berlin52 dataset

Iterations are independant

Parameters:
-- Nb of clusters: 6
-- Threshold: 8

Target distance: 7544.37

Iterations: 1000
Minimum distance: 7780.29
Maximum distance: 9624.34
Average distance: 8522.17
Standard deviation: 353.60
Execution time average: 0.096 sec
Total execution time: 98.702 sec



NOTES

The threshold must be less than or equal to the number of clusters. If not, it would involve, at a certain level, generating more clusters than cities:

Number of clusters=5
Threshold=4

--> Create 5 clusters with 4 cities=4 clusters with 1 city and 1 empty cluster

I use K-means clusters because they are simple to use, it is possible to set the number of clusters and it is based on basic geographic data (e.g. DBSCAN clusters use density and noise)

This algorithm can solve basic TSP: it only uses point coordinates, and does not take into account constraints between points (e.g. 2 cities that cannot be connected or prohibited areas) or earth curve

LIMITATION

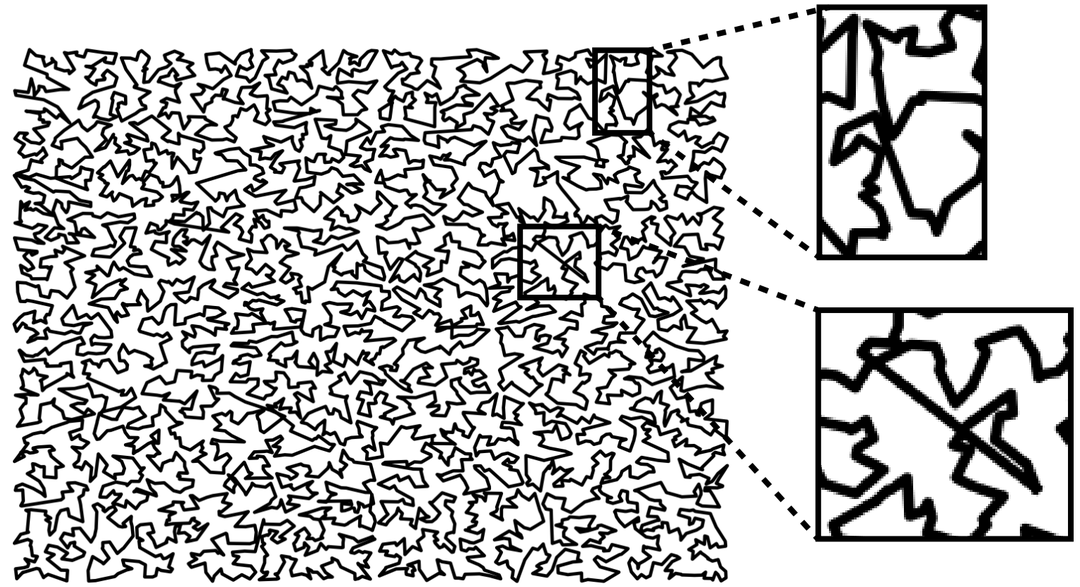
Partitioning:
Clusters do not share information with each other. That involves they do not take “connection” path into account to get the shortest path.

Centroids:
Using cluster centroids to determine the last points (in clusters, to compute perfect routes) can skew the result due to a large number of cities and the scattering

First != Last:
First and last points can be the same (same point closest to the previous and next cluster) when computed with my algorithm.

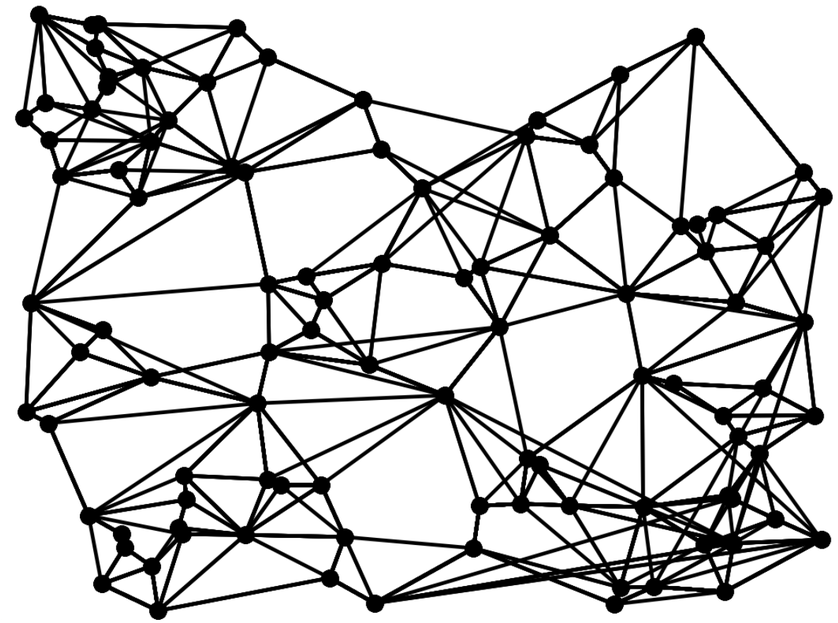
Non-uniform cluster:
The number of cities inside a cluster is not controlled. Therefore, clusters can be non-uniform (large number of cities gap between clusters)

LIMITATION



Error due to non-uniform cluster sizes

Poor repeatability



Routes computed for the same set of cities

NEXT STEPS

-- Post processing optimization (e.g. local minimum or genetic algorithm)
-- New algorithm to get first and last points
-- Change brute force algorithm by another one (e.g. tree based search)
-- New clustering algorithm, to control cluster uniformity