"Graph Analytics"
UNIMORE course

Air-Aware Walking Routes

Air Quality Analysis and Path Finding

on Modena City

04/03/2025
Giorgia Bertacchini matr. 193871

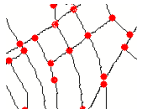
The shortest route isn't always the best!

How help people find the best path based on multiple factors, including air pollution?

The solution:

Collect real-time sensor data on air pollution.

Create an interpolated pollution map.



Model the city as a graph, where roads have also the pollution values.

Find the optimal path based on air quality instead of just distance.

A smarter, healthier way
to navigate the city!

Input Data

Sensors coordinates

ID_STATION	NAME	LONGITUDE	LATITUDE	ID_AQ_LEGAL…
1	Parco Ferrari	10.9063267…	44.6506116…	625
14	Via Alessandro Vol…	10.913416	44.645073	<null>
15	Via Pavia	10.930801	44.624152	<null>
16	Piazza Manzoni	10.929349	44.6344	<null>
17	Piazza Grande	10.92611	44.646247	<null>
18	Via Giardini	10.9047394…	44.6360271…	627
19	Dipartimento di in…	10.948936	44.629214	<null>
20	Via Villa d Oro	10.945775	44.658092	<null>
22	Policlinico di Mod…	10.943549	44.634784	<null>
23	MASA	10.9299136	44.6574023	<null>

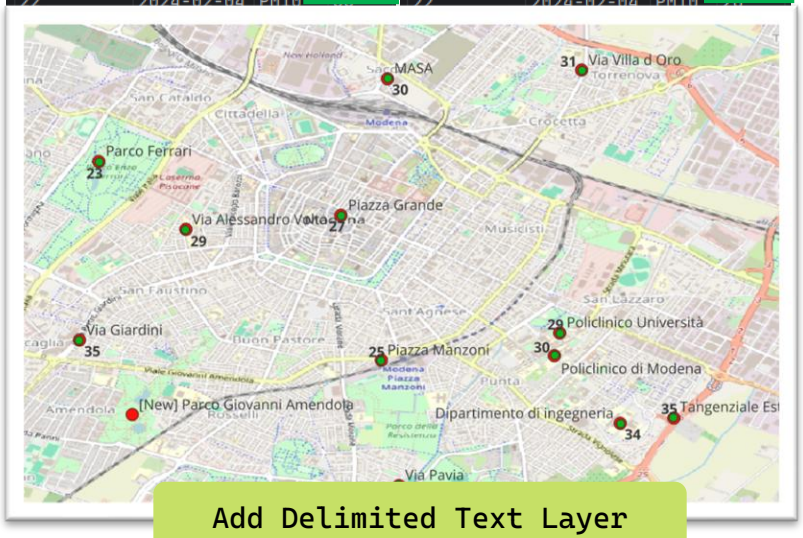
Add Delimited Text Layer

Sensors PM10 values (smoothed)

ID_STATION	DATE	PARA	VALUE
1	2024-02-04	PM10	23
14	2024-02-04	PM10	29
15	2024-02-04	PM10	24
16	2024-02-04	PM10	25
17	2024-02-04	PM10	27
18	2024-02-04	PM10	35
19	2024-02-04	PM10	34
20	2024-02-04	PM10	31

Sensors PM10 values (sharped)

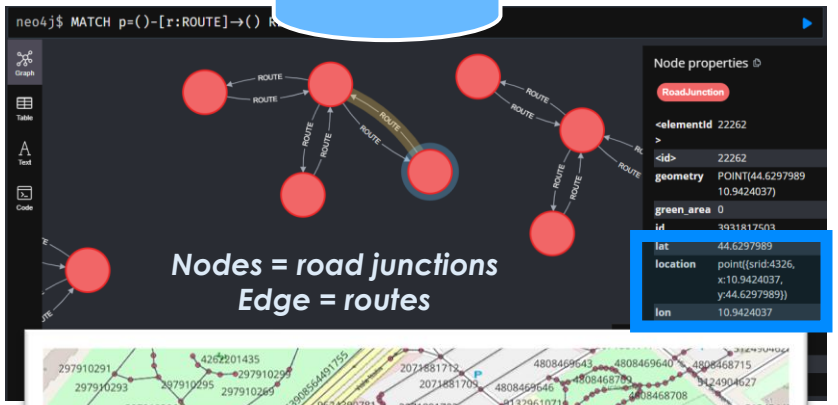
ID_STATION	DATE	PARAM	VALUE
1	2024-02-04	PM10	13
14	2024-02-04	PM10	31
15	2024-02-04	PM10	16
16	2024-02-04	PM10	43
17	2024-02-04	PM10	18
18	2024-02-04	PM10	49
19	2024-02-04	PM10	50
20	2024-02-04	PM10	51

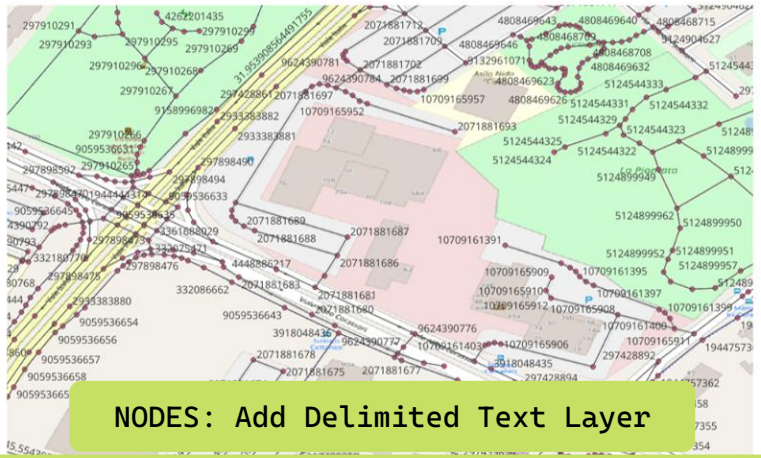


Add Delimited Text Layer

neo4j

City footpath





NODES: Add Delimited Text Layer

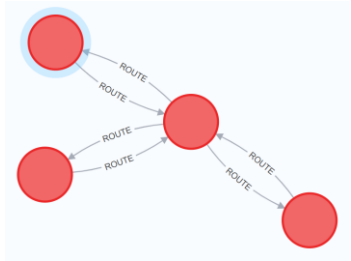
EDGES: Geometry with expression:

```
make_line(make_point("source_lon", "source_lat"),
make_point("target_lon", "target_lat"))
```

Data Analysis of

Modena footpath graph

neo4j



RoadJunction node: ***Longitude & Latitude***

```
MATCH (n:RoadJunction) RETURN
min(n.lon) as min_lon, max(n.lon) as max_lon,
min(n.lat) as min_lat, max(n.lat) as max_lat
```

min_lon	max_lon	min_lat	max_lat
10.8623693	10.9887539	44.6009373	44.6908201

The width and height of the IDW interpolation **raster** are given by the **minimum** and **maximum coordinates** of the Modena road junctions (with 5% extended limits).

ROUTE edge: ***Green_area***

It can be seen that many roads do **not have green areas** (equal to 0). This is a variable we can consider when looking for the optimal routes.

```
MATCH (s:RoadJunction)-[r:ROUTE]->(d:RoadJunction)  WITH
sum(CASE WHEN r.green_area = 0 THEN 1 ELSE 0 END) AS count_0_green_area,
sum(CASE WHEN r.green_area > 0 THEN 1 ELSE 0 END) AS count_with_green_area
RETURN count_0_green_area, count_with_green_area
```

count_0_green_area	count_with_green_area
77714	8852

```
MATCH (s:RoadJunction)-[r:ROUTE]->(d:RoadJunction)
RETURN DISTINCT r.green_area
```

r.green_area
0
100
50

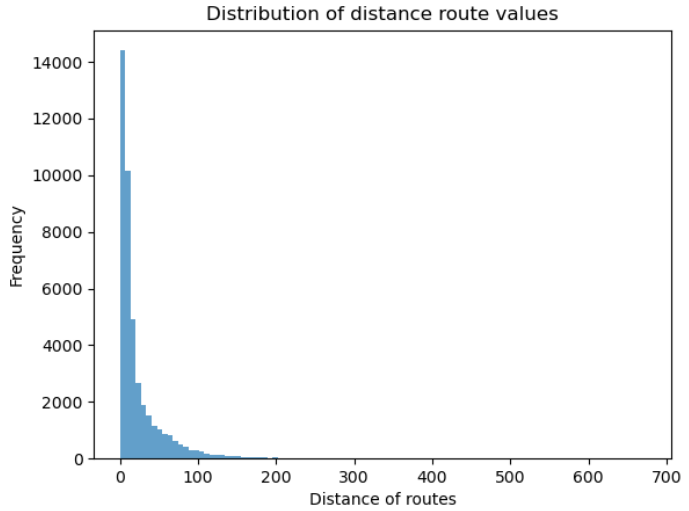
ROUTE edge: ***Distance***

```
MATCH (s:RoadJunction)-[r:ROUTE]->(d:RoadJunction)
RETURN
  max(r.distance) AS max_distance,  min(r.distance) AS min_distance,
  avg(r.distance) AS avg_distance,  stdev(r.distance) AS stdev_distance
```

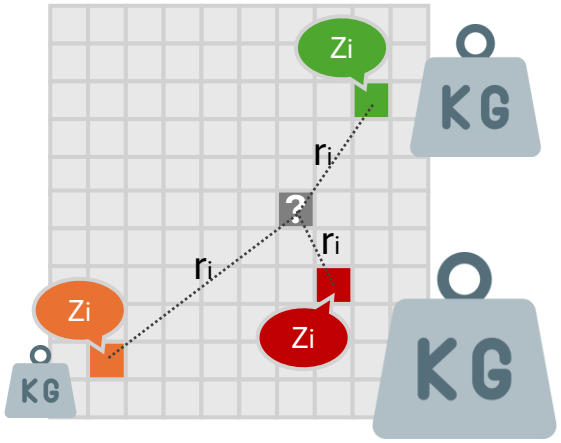
max_distance	min_distance	avg_distance	stdev_distance
672.21881434655	0.08402824390788019	24.710080175546477	37.054355228573726

The **range** of distances is quite broad, but most roads have a distance of less than 100 meters.

However, this is a value that should be considered when adding the **PM10 value**, as we want to minimize the time spent on roads with high PM10 levels.

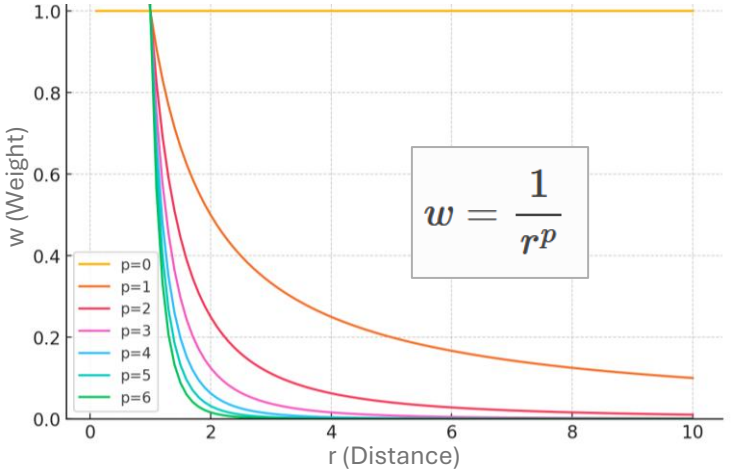


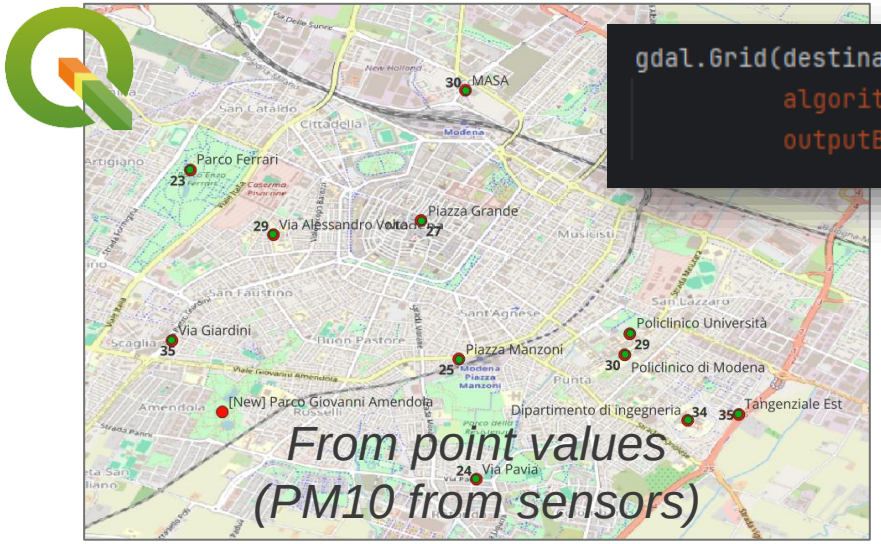
IDW Raster



IDW (Inverse Distance Weighting) is a **spatial interpolation** method that estimates a point's value by weighting nearby known points based on **distance**.

$$Z = \frac{\sum_{i=1}^{n} \frac{Z_i}{r_i^p}}{\sum_{i=1}^{n} \frac{1}{r_i^p}}$$

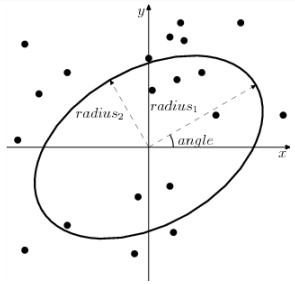


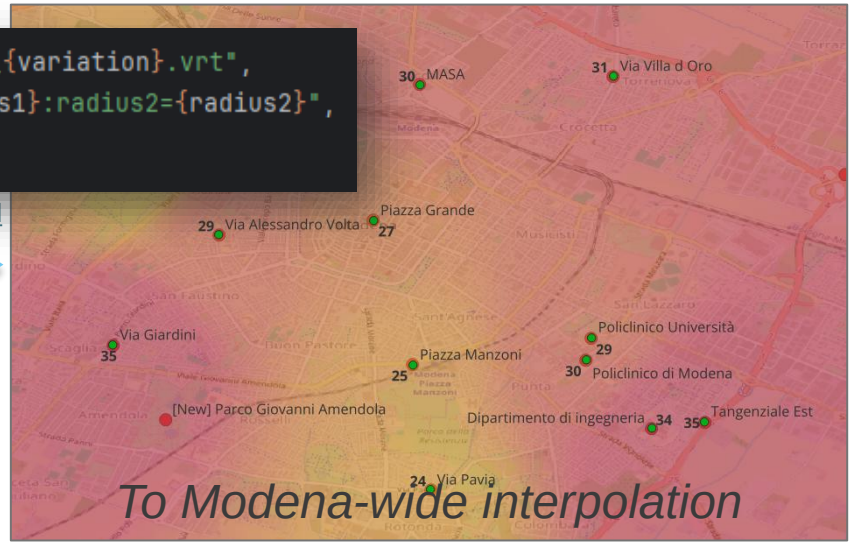


GDAL

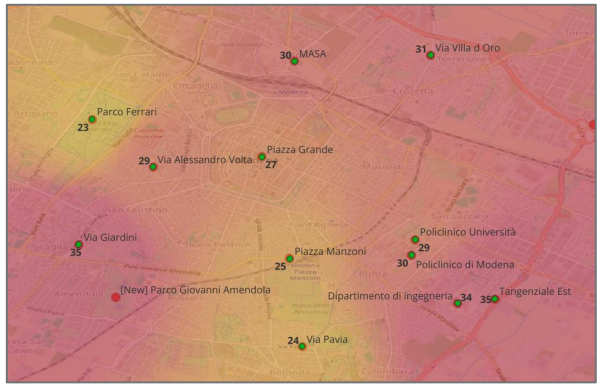
```
gdal.Grid(destination_path, srcDS: f"./output/sensors/meas_{variation}.vrt",
          algorithm=f"invdist:power={power}:radius1={radius1}:radius2={radius2}",
          outputBounds=[x_min, y_min, x_max, y_max])
```

https://gdal.org/en/stable/tutorials/gdal_grid_tut.html

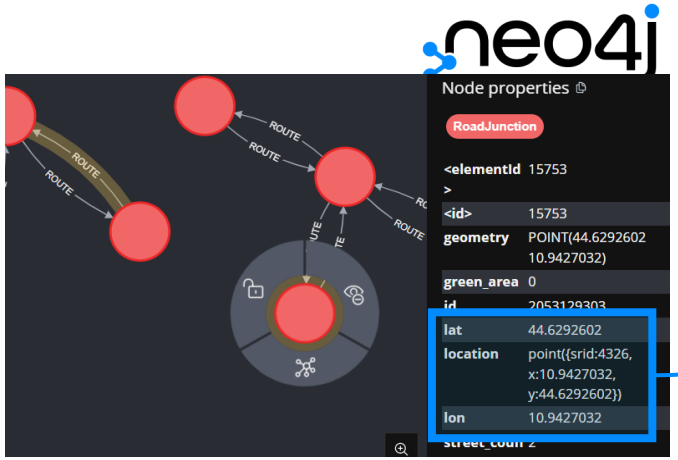




Integration with the Footpath Network

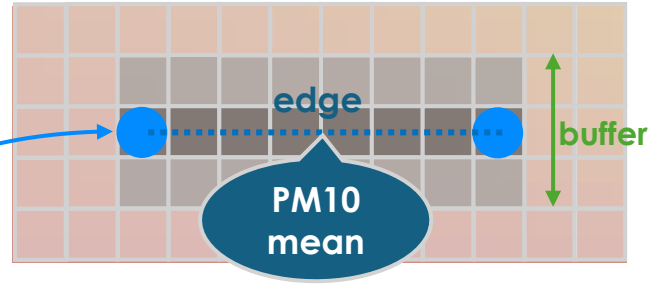


PM10 Raster



Road junction coordinates

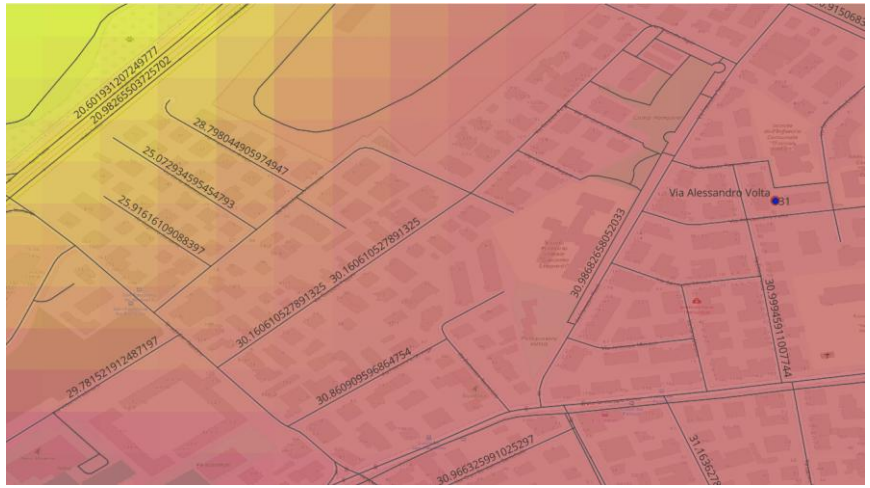
lat, lon



To reduce the number of database calls and improve operation time, I used a **single Cypher query** to efficiently load the PM10 value as an edge property in bulk.

```
UNWIND $pairs AS pair
 MATCH (s:RoadJunction)-[r:ROUTE]->(d:RoadJunction)
 WHERE s.id = pair.source AND d.id = pair.destination
 SET r.pm10 = pair.mean_air_quality
 WITH s, d, pair
 MATCH (d)-[r2:ROUTE]->(s)
 SET r2.pm10 = pair.mean_air_quality
 RETURN pair.mean_air_quality
```

opposite direction



Roads with PM10 value

Path finding

Relationship weight property

Distance

Simple search for the shortest route.

Distance: is a property of the **ROUTE** edge, measured in meters.

PM10 exposure of route

$$PM10_{\text{route}} = PM10 \cdot d$$

PM10: is the average value previously calculated on the raster along the entire edge.

d: for a value of total exposer to PM10 along the route.

Green area of route

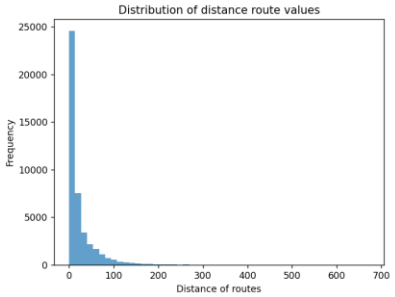
$$GA_{\text{route}} = \frac{d}{\frac{greenArea}{100} + \epsilon}$$

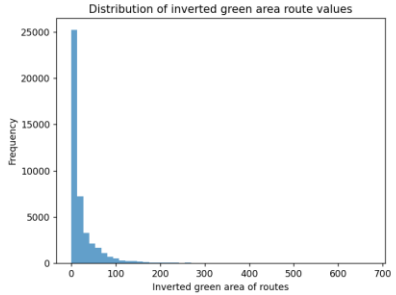
inverse: we need a property that the algorithm minimizes

ϵ is a small value (e.g. 1) to avoid division by zero.

/100: to reduce the impact of the presence of green area.

d: for a value related to the route distance with green area.





Combined weight

$$normPM10_{\text{route ij}} = \frac{PM10_{\text{route ij}} - PM10_{\text{route min}}}{PM10_{\text{route max}} - PM10_{\text{route min}}}$$

$$normGA_{\text{route ij}} = \frac{GA_{\text{route ij}} - GA_{\text{route min}}}{GA_{\text{route max}} - GA_{\text{route min}}}$$

	PM10 routes	Green area routes
max	~20712.293	~672.219
min	~1.900	~0.084

$$\text{weight}_{ij} = w_{\text{pm10}} \cdot normPM10_{\text{route ij}} + w_{\text{ga}} \cdot normGA_{\text{route ij}}$$

Normalization: To make the variables comparable, it is essential to account for differences in their **units** and **scales**, ensuring consistency across measurements.

Weights: assigned to each component make it possible to attribute varying **degrees of importance** to them, ensuring a more flexible and detailed evaluation.

Path finding

Path finding algorithms

Dijkstra Source-Target Shortest Path

"supports weighted graphs with positive relationship weights."

A* Shortest Path

"is an informed search algorithm as it uses a heuristic function to guide the graph traversal."

Yen's Shortest Path

"computes a number of shortest paths between two nodes."

```
MATCH (source:Station {name: 'Kings Cross'}), (target:Station {name: 'Kentish Town'})
CALL gds.shortestPath.astar.stream('myGraph', {
    sourceNode: source,
    targetNode: target,
    latitudeProperty: 'lat
    longitudeProperty: 'lo
    relationshipWeightProp
})
YIELD index, sourceNode, t
RETURN
    index,
    gds.util.asNode(source
    gds.util.asNode(target
    totalCost,
    [nodeId IN nodeIds | g
    costs,
    nodes(path) as path
ORDER BY index
```

```
MATCH (source:Location {name: 'A'}), (target:Location {name: 'F'})
CALL gds.shortestPath.dijkstra.stream('myGraph', {
    sourceNode: source,
    targetNodes: target,
    relationshipWeightProperty: 'cost'
})
YIELD index, sourceNode, targetNode, totalCost, nodeIds, costs, path
RETURN
    index,
    gds.util.asNode(sourceNode).name AS sourceNodeName,
    gds.util.asNode(targetNode).name AS targetNodeName,
    totalCost,
    [nodeId IN nodeIds | gds.util.asNode(nodeId).name] AS nodeNames,
    costs,
    nodes(path) as path
ORDER BY index
```

https://neo4j.com/docs/graph-data-science/current/algorithms/pathfinding/

Example

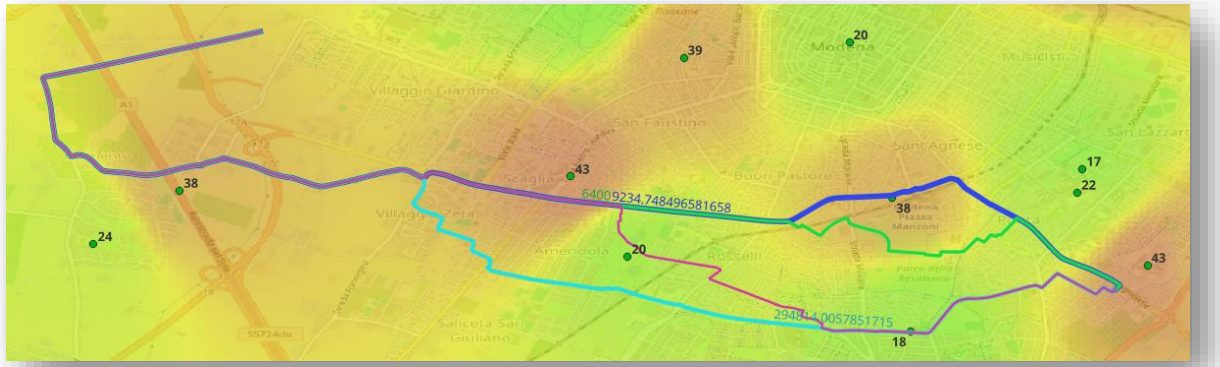
LEGEND

Distance (value: total distance)

Route PM10 (value: total PM route)

Route Green area (value: total green area)

Combined weight



	Shortest Distance	Lowest PM10	Highest Green Area	Balanced (PM10 + Green area)
Distance [m]	~ 9234.748	~ 9810.031	~ 9539.620	~ 9784.344
Total PM10 route [µg]	~ 310324.819	~ 294817.006	~ 316246.106	~ 297954.816
Total green area	0	0	6400	3800

Code structure

https://github.com/giorgiaBertacchini/graph_routing_airquality

pre_analysis.py
analyze real pm10 data from ARPAE sensors in Modena, to create synthetic data.

Sensors coordinates

Sensors PM10 values

CSV

neo4j

City footpath

graph_bridge.py
It contains the class that communicates directly with the graph database. It takes care of connecting to the database and sending cypher queries.

```
CALL gds.shortestPath.dijkstra
| astar | yeans.stream
```

```
MATCH (s)-[r:ROUTE]->(d)
RETURN ...
```

```
MATCH (s)-[r:ROUTE]->(d)
RETURN s.lon, s.lat, d.lon, d.lat
```

```
MATCH (s)-[r:ROUTE]->(d)
SET r.combined_weight = ...
```

```
MATCH ()-[r:ROUTE]->()
SET r.pm10
```

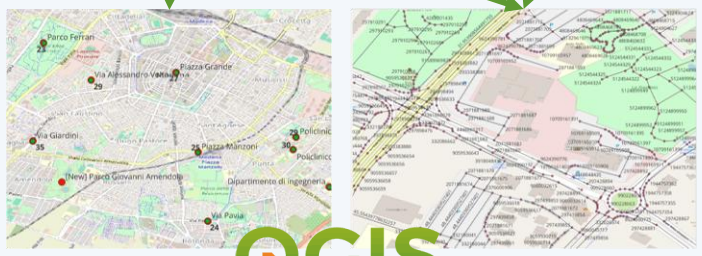
main.py

export_to_csv.py
Create CSV files with sensors coordinates and pm10 values, or with road junctions info and routes (edges) info, to visualize them on a map.

sensors /data .csv

road_ junctions .csv

edges .csv



QGIS

interpolation.py
With coordinates and PM10 values create through IDW interpolation method a Modena-wide map of air-quality.

.tif

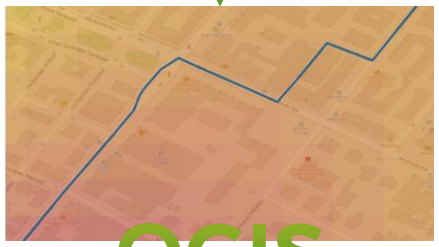
QGIS

merge_airquality_ footpath.py
With the coordinates from Neo4j, calculates the PM10 mean along the each route.

QGIS

footway_routing.py
Reads the required routing configurations from a JSON file, creates additional properties on the edges (e.g. effective_PM10, combined_weight) and via a path finding algorithm returns the best path(s).

.geojson



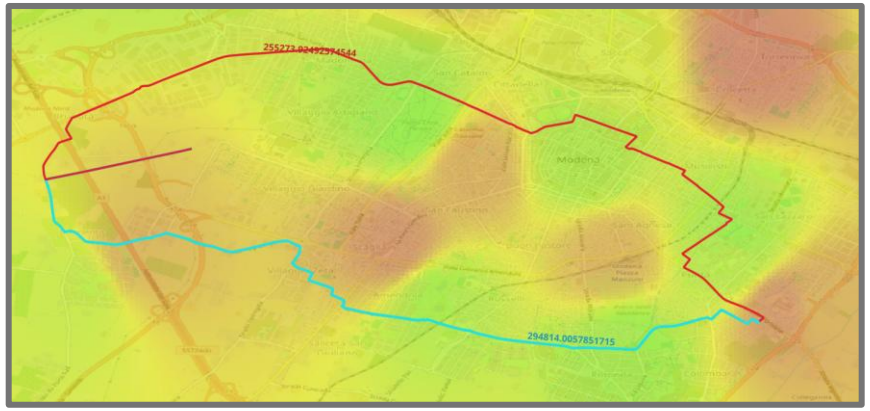
QGIS

Routing Results

Comparing routes with PM10 values:
'sharped' (deviation standard of 20) and 'smoothed' (deviation standard of 10)

From these tests, we can observe that scenarios with sharper variations in PM10 levels and those with more uniform PM10 distributions can lead to different optimal routes, depending on air pollution patterns.

	Total distance [m]
Sharped	~11744.035
Smoothed	~9810.031



LEGEND

Sharped (value: total route PM10)

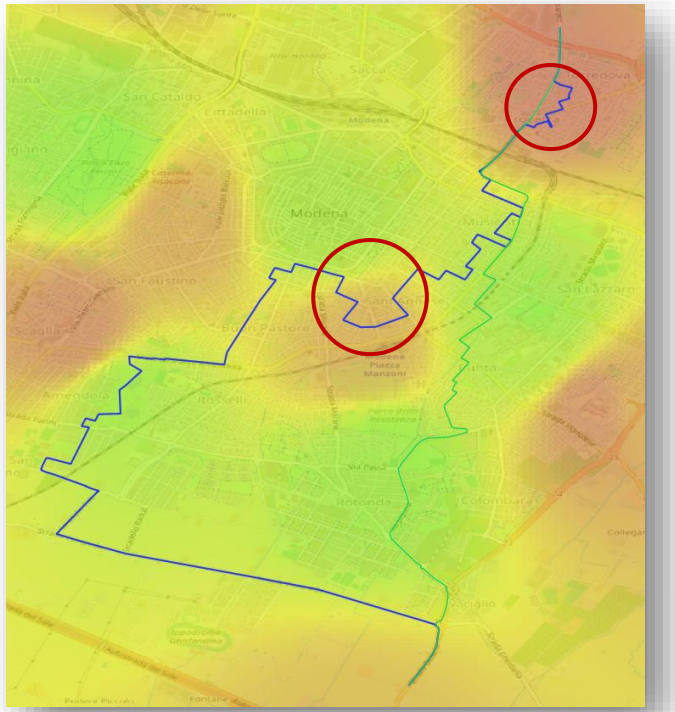
Smoothed (value: total route PM10)

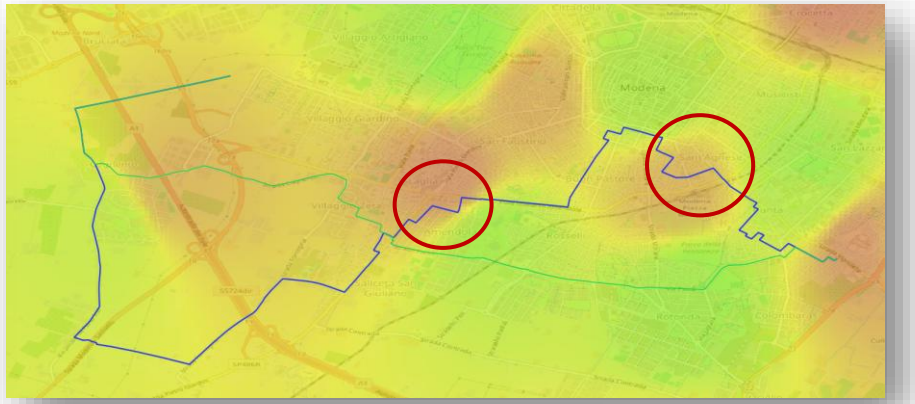
	Total distance [m]
Sharped	~6977.433
Smoothed	~5982.913

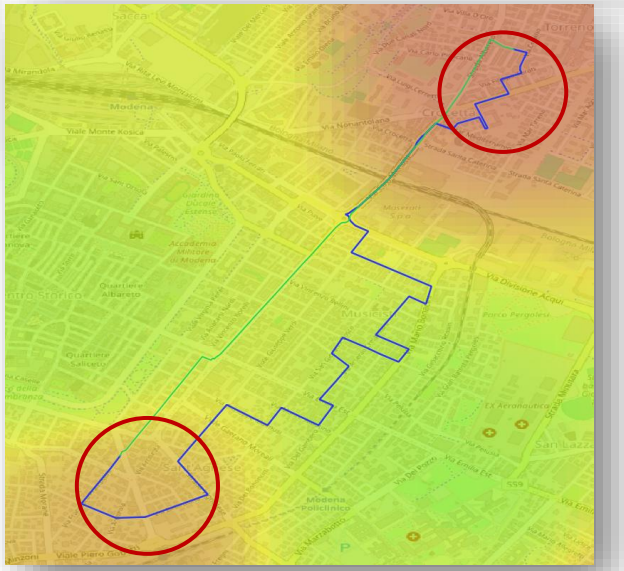
	Total distance [m]
Sharped	~7765.125
Smoothed	~7656.867

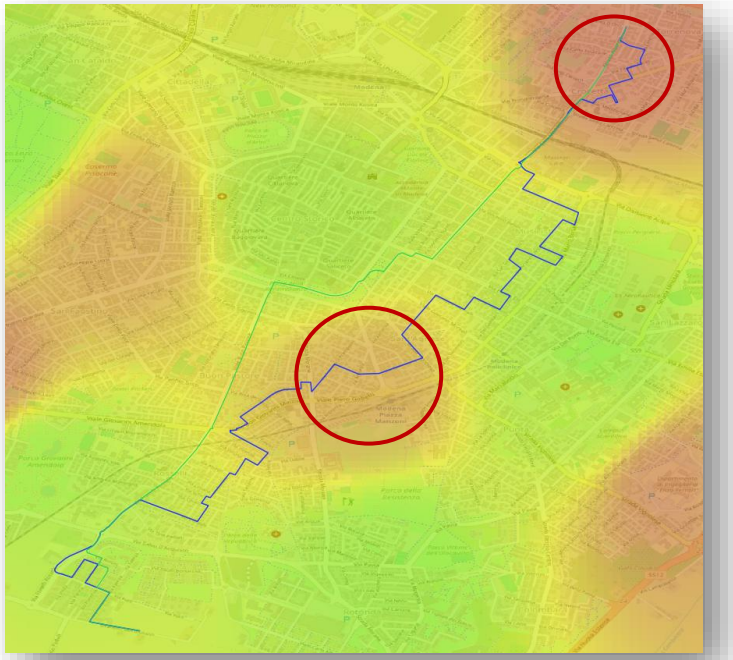
Routing Results

Comparing routes with PM10 multiplied or divided by the distance of the route









LEGEND

(r.pm10) * (r.distance)

(r.pm10) / (r.distance)

From the tests I conducted, it seems that when the algorithm calculates the optimal route based on the PM10 × distance value, the results are more accurate and consistent.

On the other hand, paths calculated based on the PM10/distance value seem more irregular, often choosing to unnecessarily pass through more polluted areas.

Routing Results

Comparing paths between Dijkstra and A algorithm*

LEGEND

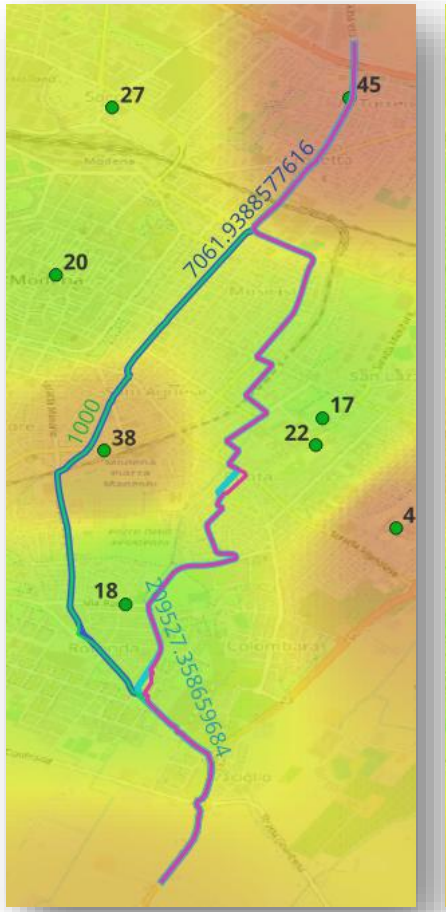
Distance (value: total distance)

PM10 route (value: total PM route)

Green area route (total green area)

Combined weight

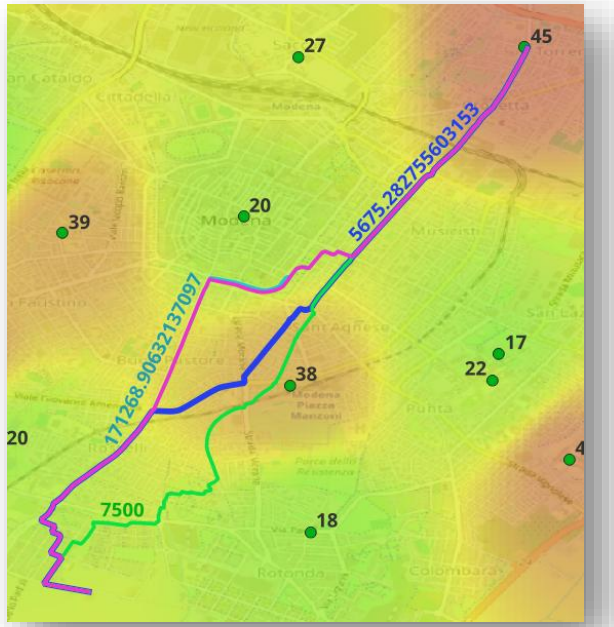
Dijkstra algorithm



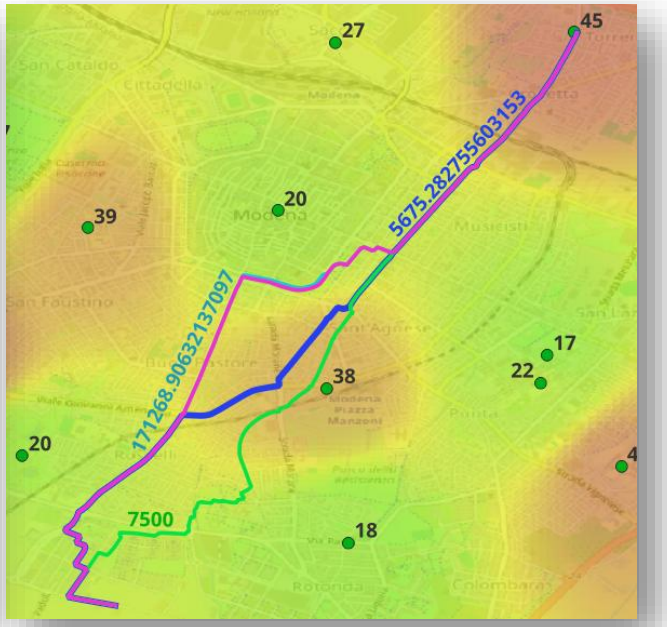
A algorithm*

From these examples, we can observe that the results provided by both algorithms — in this case, the calculated optimal paths — are exactly the same.

Dijkstra algorithm



A algorithm*



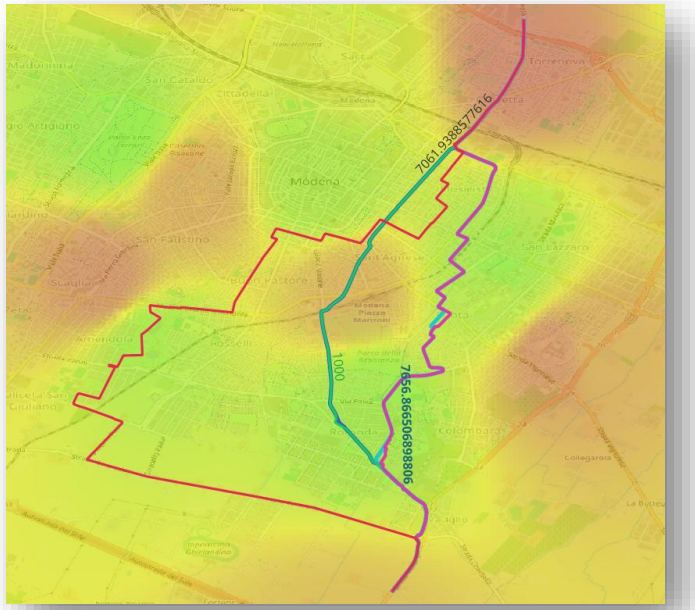
Routing Results

Evaluating whether to divide the combined weight by the distance

combined_weight = (ratio_pm10 * norm_pm10) + (ratio_greenArea * norm_greenArea)

combined_weight = (ratio_pm10 * norm_pm10) + (ratio_ greenArea * norm_greenArea)/distance

Similarly, about the combined weight of the two factors, it appears that dividing by distance results in more random routes that deviate from those suggested by the optimal path based on total PM10 and total green areas.



LEGEND

Distance (value: total distance)

PM10 route (value: total PM route)

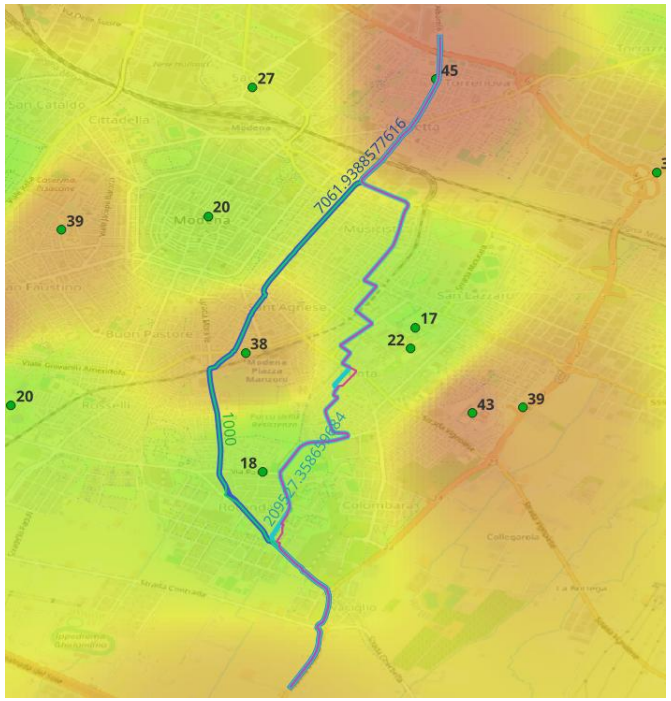
Green area route (total green area)

Combined weight

Combined weight

Routing Results

Other complete examples



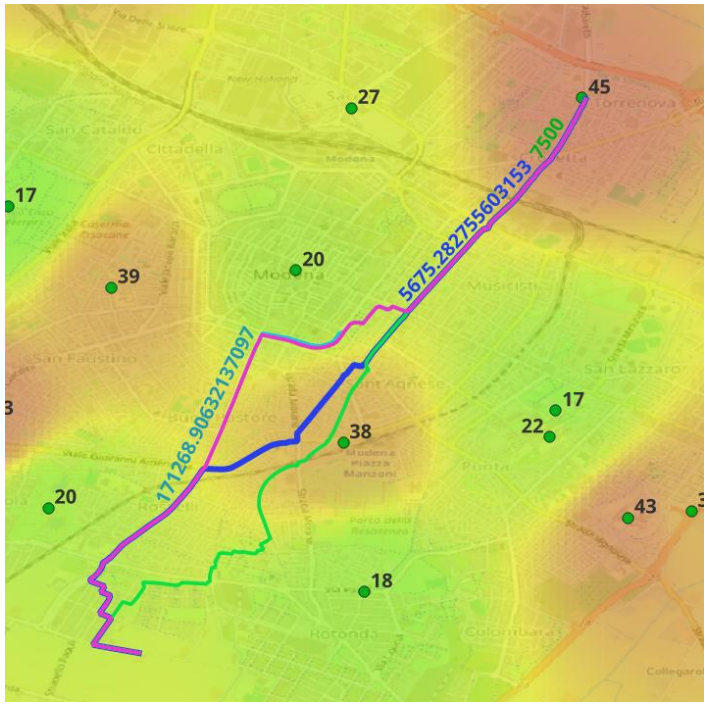
LEGEND

Distance (value: total distance)

Route PM10 (value: total PM route)

Route Green area (value: total green area)

Combined weight



	Shortest Distance	Lowest PM10	Highest Green Area	Balanced (PM10 + Green area)
Distance [m]	~ 7031.939	~ 7656.867	~ 7085.069	~ 7704.265
Total PM10 route [µg]	~ 218838.276	~ 209527.359	~ 219263.394	~ 210350.680
Total green area	0	0	1000	1000

	Shortest Distance	Lowest PM10	Highest Green Area	Balanced (PM10 + Green area)
Distance [m]	~ 5675.283	~ 5982.913	~ 5858.856	~ 5982.164
Total PM10 route [µg]	~ 174925.483	~ 171268.906	~ 181982.648	~ 171601.845
Total green area	0	2100	7500	3900


Thank You for Your Attention!