GRAFMOVE: Graph-based Mobility Optimization and Visualization Engine

Federica Rollo
Enzo Ferrari Engineering Department
University of Modena and Reggio Emilia
Modena, Italy
federica.rollo@unimore.it

Laura Po
Enzo Ferrari Engineering Department
University of Modena and Reggio Emilia
Modena, Italy
laura.po@unimore.it

Abstract

Traditional pedestrian routing systems prioritize finding the shortest path, often neglecting user preferences and contextual factors such as green spaces or safety concerns. To address this limitation, we present GRAFMOVE, a customizable routing system that leverages Neo4j and OpenStreetMap data to integrate dynamic, user-specific criteria into pathfinding. GRAFMOVE constructs a footpath graph enriched with contextual data and implements a flexible cost function, enabling users to balance path length with personalized factors. The system includes an interactive dashboard for real-time route visualization and optimization. We demonstrate GRAFMOVE's capabilities through case studies, including Point Of Interest-based routing for tourists and personalized green path recommendations. Our approach advances pedestrian routing by offering a scalable, adaptable solution.

CCS Concepts

• **Information systems** → *Information systems applications.*

Keywords

Pedestrian Routing, Personalized Navigation, Route Visualization

ACM Reference Format:
Federica Rollo and Laura Po. 2025. GRAFMOVE: Graph-based Mobility Optimization and Visualization Engine. In *The 33rd ACM International Conference on Advances in Geographic Information Systems (SIGSPATIAL '25), November 3–6, 2025, Minneapolis, MN, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3748636.3762798

1 Introduction

Pedestrian routing systems have traditionally focused on identifying the shortest path between two points, overlooking the diverse preferences and contextual needs of pedestrians. As an example, pedestrians may prefer longer routes that avoid trafficked roads to reduce air pollution exposure, or they may prioritize paths that traverse green spaces or pass by Points of Interest (POIs), such as historical landmarks. Additionally, time-dependent factors, such as street lighting conditions, can significantly influence route preferences, particularly during evening hours when safety and visibility become paramount. These considerations highlight the limitations of conventional routing algorithms and commercial tools (e.g., Google Maps, OsmAnd), which do not integrate user-specific dynamic constraints into path finding processes. For example, OpenRouteService[1] and VROOM [3] implement POI-aware navigation but optimize primarily for distance or time. CAPRIO [2] integrates environmental criteria to minimize outdoor exposure but remains constrained to single-objective formulations.

In this paper, we demonstrate GRAFMOVE, a graph-based system designed to address the challenges of personalized pedestrian routing, building on recent advances in graph-based path finding algorithms [4–6]. GRAFMOVE represents a novel approach that allows users to dynamically define multi-criteria routing functions, balancing path length with contextual factors such as green spaces and POI proximity through the creation of the footpath[2] graph. Furthermore, GRAFMOVE includes an interactive dashboard for visualizing both the data integrated in the graph and the resulting routes of personalized pedestrian routing. This work is part of the AIQS project [7] that aims to improve the accuracy of air quality sensors using artificial intelligence and to enable pollution-aware pedestrian routing for healthier urban mobility.

In the following, we detail the design and implementation of GRAFMOVE and demonstrate its capabilities through some case studies and visualizations. Through the demonstration, researchers and urban planners can create new footpath graphs, examine precomputed graphs and interact with the dashboard to explore urban data, identify POIs, and determine optimal routes. The source code is available at https://github.com/federicarollo/GRAFMOVE, while the demo video is available at https://vimeo.com/1093401006/5e85716382 and graph dumps are sent on request.

2 GRAFMOVE system

Our proposed GRAFMOVE system consists of a *back-end component* aimed to create the footpath graph, integrate POIs, identify green areas and find the best route between two or multiple points in the city, and a *front-end component* to visualize the footpath network and solve routing problems. Figure 1 shows the GRAFMOVE workflow: the graph is created in Neo4j based on OSM data, while the visualization is implemented using NeoDash. Since OSM provides

SIGSPATIAL '25, Minneapolis, MN, USA

ACM ISBN 979-8-4007-2086-4/2025/11
https://doi.org/10.1145/3748636.3762798

[1] https://openrouteservice.org
[2] Since in this paper we focus on paths where pedestrians are permitted, we use the term footpath broadly to refer to all walkable paths, including those not exclusively reserved for pedestrians.

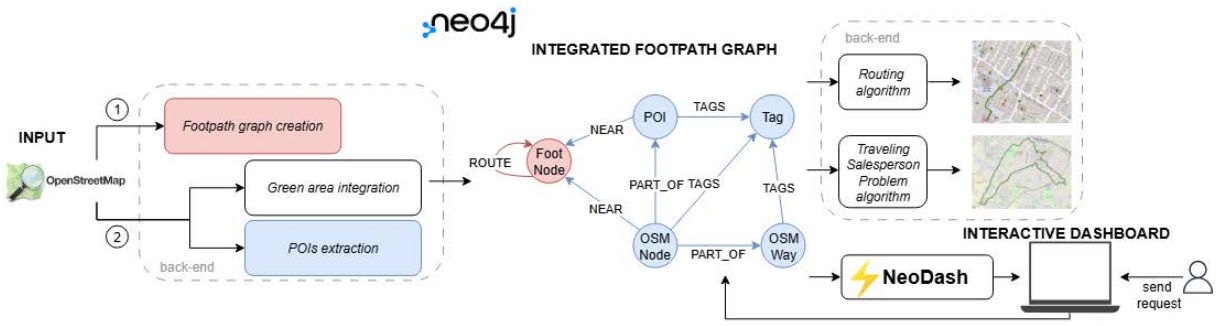


Figure 1: GRAFMOVE system

extensive global coverage of footpath data, our approach is highly flexible, scalable, and easily applicable to different cities.

2.1 Integrated footpath graph

The footpath network in OSM is structured as a collection of ways (roads) and nodes (points on the roads that define the geometric shape). Building on this representation, we formalize the footpath graph where nodes (labeled as `FootNode`) correspond to OSM nodes, interconnected via `ROUTE` edges. Each `FootNode` has a specific spatial position (latitude and longitude) and is identified by the OSM numerical code. The `ROUTE` relations store all attributes from OSM such as length, type of path (sidewalk, trail, pedestrian-only road), surface material (paved, unpaved).

After selecting the area of interest, it is necessary to define its central geographic coordinates and radius of coverage. These parameters are utilized by the OSMnx Python library [1] to extract OSM data and convert them into a graph structure stored in a file (GRAPHML format) that is then loaded in a graph database using the Neo4j Cypher query language and the APOC library.[3] To enhance scalability, periodic commit queries were implemented, ensuring efficient transaction management and reducing memory overhead during large-scale data processing. The spatial plugin[4] is integrated in Neo4j to facilitate the storage and querying of spatial data. A spatial layer is created to store the `FootNode` nodes.

2.1.1 Green area integration. In OSM, green spaces (e.g., parks, forests, and gardens) are represented through specific attributes that define their type, purpose, and characteristics. Small green areas (e.g., a single tree or a small garden) are defined as nodes, while larger areas like parks are mapped as polygons, i.e., interconnected ways that form closed boundaries to represent the spatial extent of the green area.

Our goal is to identify footpaths located within green areas. Thus, GRAFMOVE employs the Overpass API to query OSM nodes and ways associated with the attributes specified in Table 1. The retrieved nodes are compared with the `FootNode` nodes based on their OSM identifiers. If a match is found, the corresponding `FootNode` is annotated with the property *green_area='yes'*. For the ways, the GPS coordinates of their boundaries are used to delineate the perimeter of each green area. Subsequently, an additional OSM query retrieves all nodes located within these boundaries and the property *green_area='yes'* of the corresponding `FootNode` instances is assigned. Then, the property *green_area* is added to the relation `ROUTE` as follows: 100 if both the source node and the target node of the relation are in a green area, 50 if only one, 0 otherwise.

A derived property is added to the `ROUTE` relations as:

$$\text{green\_area\_weight} = \text{distance} \times \left(1 + \frac{\text{green\_area}}{100}\right)^{-1}$$

where *distance* is the length of the road. In this way, we define a weight equal to the distance if the path is not located within a green area, reduced to half the distance if the entire path lies within a green area, reduced by one-third if the path is in close proximity to a green area.

Table 1: OSM attributes for green areas and POIs integration.

	Key	Values
Green areas	landuse	grass, flowerbed, meadow, forest, vineyard, village_green, recreation_ground, orchard, nature_reserve
	leisure	garden, park, dog_park, pitch, nature_reserve, golf_course, garden
	natural	wood, tree_row, scrub, heath, grassland, fell
	barrier	hedge
POIs	amenity	all the values
	tourism	all the values
	place	square

Table 2: Key statistics of exemplar graphs and execution time.

City name	Mantova	Modena	Turin	Milan
Area (Km^2)	16	100	196	400
# `FootNode`	9967	46151	161584	559025
# `FootNode` in green areas	1113	4892	25740	60838
# `POI`	1028	2111	11618	44336
# `OSMWay`	299	983	2214	14519
# `OSMNode`	3318	10521	29194	29891
# `ROUTE`	18437	94134	306953	1105837
Execution time (minutes)				
- Graph creation	0.20	4	49	200
- Green areas integration	0.15	2	9	46
- POIs integration	0.30	3	15	75

[3] https://github.com/neo4j-contrib/neo4j-apoc-procedures
[4] https://github.com/neo4j-contrib/spatial

2.1.2 POIs extraction. POIs denote specific locations that hold particular significance or utility for a given purpose, such as restaurants, hotels, places of worship, or historical monuments. In OSM, POIs can be represented either as nodes or ways based on their nature and characteristics. We implemented two different Overpass queries to extract nodes and ways related to POIs specifying the presence of the OSM attributes listed in Table 1. For each POI represented as way, a node with two labels, i.e., `OSMWay` and `POI`, is generated in the graph. For each node contained in the way, a node with label `OSMNode` is generated and linked to the corresponding `OSMWay` node with the relation `PART_OF`. On the other hand, the POIs mapped as nodes in OSM are represented in the graph by nodes with labels `OSMNode` and `POI`.

All the attributes that describe POIs in OSM are reported in the graph through the `Tag` node. In this way, querying the `Tag` node it is possible to select POIs according to, for example, the name or the type (e.g., square, school).

Another spatial layer is generated to store the `OSMNode` nodes. Through the optimized *withinDistance* procedure of the spatial plugin, each `OSMNode` is connected to the nearest `FootNode`.

2.2 Path finding algorithm

The Neo4j APOC library provides powerful graph algorithms for path computation, including the well-known A* (A-Star) and Dijkstra. These algorithms are used in GRAFMOVE both in the back-end and in the dashboard. Given the names of the source and target POIs, the system identifies the nearest `FootNode` instances and uses them as input for A* or Dijkstra. By employing the *distance* property as the cost function of A* or Dijkstra, GRAFMOVE identifies the shortest path. Conversely, when using *green_area_weight*, the system optimizes routes through green spaces while maintaining an acceptable distance value.

Additionally, GRAFMOVE extends routing capabilities by solving the Traveling Salesperson Problem (TSP) to visit multiple points in optimal order. The TSP implementation in GRAFMOVE exploits a Cypher query to the graph and guarantees the optimality of the resulting paths. Starting from a list of POI names, after retrieving the nearest `FootNode` nodes, the optimal path (based on *distance* or *green_area_weight*) is calculated using A* for each unique pair of nodes. Then, using an expansion-based approach[5], the query constructs circular paths that visit all POIs without repetitions. Paths are evaluated by summed edge costs (*distance* or *green_area_weight*) and the minimal-cost path is selected.

2.3 Dashboard

The visualization engine of GRAFMOVE is implemented through NeoDash, which exploits direct Cypher queries to the graph and enables interactive visualizations of the results. After installing NeoDash and setting the graph database connection, NeoDash's user interface facilitates the intuitive definition of custom visualizations through a declarative approach. Upon creation, each dashboard configuration is persistently stored as a dedicated node within the graph database. Thus, graph dumps contain also the dashboard. In addition, the dashboard can be exported as a standardized JSON file and imported into other graphs with the same structure.

[5]https://neo4j.com/labs/apoc/4.3/overview/apoc.path/apoc.path.expandConfig/

Our dashboard allows for the visualization of a graph overview (e.g., number of nodes/edges, most central nodes, POI location). In addition, routing and TSP are implemented following the same approach explained in Section 2.2 and the query results (i.e., the optimal paths) are shown on a map.

3 Demonstration scenarios

The demonstration scenarios show the ability of GRAFMOVE to create a new graph, provide useful visualizations, and identify optimal personalized routes in real-world applications.[6]

3.1 POI-based and personalized routing

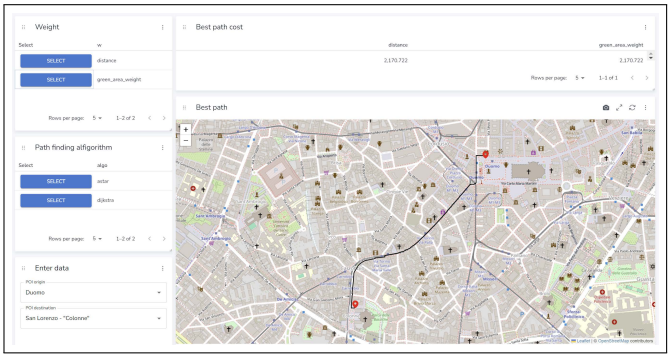
During the demonstration, participants will take advantage of the precomputed graphs (see Table 2) to generate personalized pedestrian routes that prioritize proximity to POIs. This functionality is particularly useful for tourists who wish to explore a city while visiting key attractions. Users can choose to investigate this functionality through the execution of Python code of the back-end component of GRAFMOVE or the interaction with the dashboard.

Running the ***routing.py* file**, all the tourist attractions (`POI` nodes with tags tourism='attraction', amenity='place_of_worship', place='square' or amenity='fountain') will be selected in the graph and the nearest `FootNode` to the `POI` nodes will be extracted. Alternatively, users can provide a list of two or more `FootNode` OSM identifiers as input. Users can choose to minimize the distance (if *distance* is specified in parameter weight) or find a greener alternative, i.e., a route that balances path length with the preference for traversing green areas (if *green_area_weight* is used). The script calculates the best path in less than 1 second and returns a csv file with the best path between each pair of points represented as sequence of `FootNode` identifiers and a matrix with the cost (*distance* or *green_area_weight*) of each path.

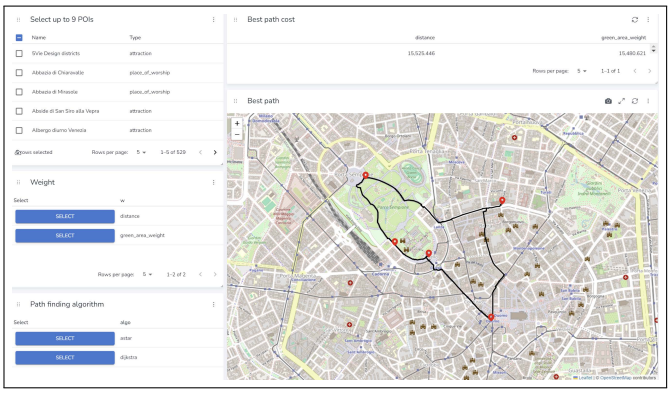
To find a closed route for visiting multiple points in the city, users can execute the ***tsp.py* file** that takes the same input of *routing.py*. The script will create a map with the optimal path and store the path in a csv file. As it relies on a Cypher query, the execution time of this script increases exponentially as the number of points grows. The script takes less than 40 seconds for up to 9 points, but from 10 points onward, the time increases significantly (8 minutes for 10 points), as does the memory usage.

Using the dashboard, users can enter the names of two POIs as origin and destination, select the routing algorithm (A* or Dijkstra), and choose the cost function (*distance* or *green_area_weight*), as shown in Figure 2.a and in the demo video. In around 5 milliseconds, GRAFMOVE automatically maps POIs to the nearest `FootNode` in the graph, computes and visualizes the optimal path on the map. Participants will interact with the dashboard by changing the selected cost function and observing in real-time how this affects the computed paths. The dashboard updates dynamically in a few milliseconds, providing visual feedback. Then, the scenario will become more complex, allowing the user to select up to 9 POIs (to ensure low execution times) from a predefined list, as shown in Figure 2.b and in the demo video. The list is obtained by dynamically

[6]GRAFMOVE is implemented in Python3.13.2 and demonstrated on a laptop powered by 32GB RAM and 14 processors with Intel(R) Core(TM) Ultra 7 155U CPU, 1700 Mhz, setting Neo4j JVM heap max size to 1G.



(a)



(b)

Figure 2: GRAFMOVE interactive dashboard.

querying the graph to select the tourist attractions. Users will also choose the routing algorithm and the cost function. The dashboard will then automatically compute and visualize the optimal route connecting all selected POIs on the map in a few seconds.

3.2 Graph creation and visualization

During the demonstration, users will be able to create a new graph. Firstly, users need to choose an area where footpaths are supposed to be and create a Neo4j database. Then, they can run the ***create_footpath_graph.py* file**, providing as input the GPS coordinates of the central point of the selected area and a radius of coverage r. GRAFMOVE will create the graph with footpaths of an area of dimension $(r*2)^2$. The script will calculate some statistics on the graph that will be shown in the standard output. Among these values, it is important to pay attention to the number of connected components and ensure that there is a component with the majority of nodes, as this will be the component used for routing. Green areas and POIs will be integrated through ***integrate_green_area.py*** and ***add_points_of_interest.py***, respectively.

The execution time depends on the dimension of the area (see Table 2). Participants can create a graph related to a small area (up to 25 Km^2) in less than one minute. Then, we guide users through the process of loading the dashboard into the new graph from a JSON file in a few seconds. In this way, users will be able to execute the scenario described in Section 3.1, on the newly constructed graph.

4 Conclusions

In this paper, we demonstrated GRAFMOVE, an interactive system for real-time mobility optimization and visualization. By leveraging graph-based modeling, GRAFMOVE supports urban planners, researchers, and tourists in transforming raw spatial data into actionable mobility insights. Furthermore, the system integrates multi-criteria path finding within Neo4j's graph traversals, enabling dynamic route recommendations that balance efficiency and comfort and going beyond traditional routing systems. Future work will focus on extending GRAFMOVE with other factors that influence the choice of pedestrian routes to implement a multi-objective optimization. A preliminary study was conducted in [8].

Acknowledgments

This work was conducted as part of the AIQS project (AI-enhanced Air Quality Sensor for Optimizing Green Routes, code: `DIP_AIQS_PO-_2025_PNRR_ECOS _SK4AF_E93C22001100001`. AIQS was funded through a closed call within the initiative "Ecosystem for Sustainable Transition in Emilia-Romagna" (ECOSISTER), financed under the National Recovery and Resilience Plan (PNRR) - Mission 4 "Education and Research", Component 2 "From Research to Business", Investment 1.5 "Creation and strengthening of innovation ecosystems, building territorial R&D leaders" - funded by the European Union - NextGenerationEU (Grant Agreement No. 0001052, dated 23/06/2022 - Project ECS_00000033 - CUP E93C22001100001).

References

[1] Geoff Boeing. 2017. OSMnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks. *Computers, Environment and Urban Systems* 65 (2017), 126–139. doi:10.1016/j.compenvurbsys.2017.05.004

[2] Constantinos Costa, Xiaoyu Ge, and Panos K. Chrysanthis. 2019. CAPRIO: graph-based integration of indoor and outdoor data for path discovery. *Proc. VLDB Endow.* 12, 12 (Aug. 2019), 1878–1881. doi:10.14778/3352063.3352089

[3] Julien Coupey, Jean-Marc Nicod, and Christophe Varnier. 2024. *VROOM v1.14, Vehicle Routing Open-source Optimization Machine.* Verso (https://verso-optim.com/), Besançon, France. http://vroom-project.org/.

[4] Christina Ludwig, Sven Lautenbach, Eva-Marie Schömann, and Alexander Zipf. 2021. Comparison of Simulated Fast and Green Routes for Cyclists and Pedestrians. In *11th International Conference on Geographic Information Science, GIScience 2021, September 27-30, 2021, Poznań, Poland (Virtual Conference) - Part II (LIPIcs, Vol. 208)*, Krzysztof Janowicz and Judith Anne Verstegen (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 3:1–3:15. doi:10.4230/LIPICS.GISCIENCE.2021.II.3

[5] Tessio Novack, Zhiyong Wang, and Alexander Zipf. 2018. A System for Generating Customized Pleasant Pedestrian Routes Based on OpenStreetMap Data. *Sensors* 18, 11 (2018), 3794. doi:10.3390/S18113794

[6] Keisuke Otaki, Ai Nakada, Tomosuke Maeda, Takayoshi Yoshimura, and Hiroyuki Sakai. 2023. Roaming Navigation for Pedestrians (Demo Paper). In *Proceedings of the 31st ACM International Conference on Advances in Geographic Information Systems* (Hamburg, Germany) *(SIGSPATIAL '23)*. Association for Computing Machinery, New York, NY, USA, Article 74, 4 pages. doi:10.1145/3589132.3625624

[7] Laura Po, Federica Rollo, Martina Casari, Matteo Angelinelli, Giorgio Pedrazzi, Roberta Turra, Chiara De Pascali, and Luca Nunzio Francioso. 2025. Enhancing Low-Cost Air Quality Sensors with AI for Smart Green Routing. In *Proceedings of the 20th Conference on Computer Science and Intelligence Systems, FedCSIS 2025, Kraków, Poland, September 14-17, 2025 (Annals of Computer Science and Information Systems)*, Marek Bolanowski, Maria Ganzha, Leszek A. Maciaszek, Marcin Paprzycki, and Dominik Slezak (Eds.).

[8] Federica Rollo and Laura Po. 2025. MODyPer: Multi-Objective Dynamic Personalized Route Planning for Vulnerable Road Users. In *Proceedings of the 33rd ACM International Conference on Advances in Geographic Information Systems* (Minneapolis, MN, USA) *(SIGSPATIAL '25)*. Association for Computing Machinery, New York, NY, USA. doi:10.1145/3748636.3764160