Route generation for minimal airborne particulate exposure for walking and cycling trips in London

Madeleine Itzel Santiago Estrada

Master of Science

Data Science

School of Informatics

University of Edinburgh

2023

Abstract

The rising atmospheric pollutant levels, primarily in urban centres, present alarming health risks, including cardiopulmonary and cerebrovascular diseases [2]. While urban navigation tools abound, a gap exists in platforms considering pollutant exposure. Therefore, the aim of this project is the development of an Android application to address this void. In this regard, the mobile application emphasises generating routes with minimal $PM_{2.5}$ pollutants, both for cyclist and pedestrian roads, focused on London's central area.

With this goal, two core research questions are addressed: (1) The route with the least airborne pollutant exposure, and (2) a route with a set Target Distance (D_t) also minimizing pollutant exposure. On this subject, the first question aligns with a Pathfinding problem, whereas the second one pertains to an Arc-Orienteering one. In order to answer these two problems, two algorithms were explored, Dijkstra's Algorithm and a Two-Phased Best Nearest Neighbour. In addition, for the routes going from point A to point B, a middle-ground solution was proposed using a Weighted Dijkstra approach, in the cases where distance from the cleanest and shortest routes showed significant disparity.

Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Madeleine Itzel Santiago Estrada*)

Acknowledgements

I would first like to express my sincere gratitude to Professor D.K. Arvind for his guidance and supervision of this work.

I would like to thank my parents, Geraldina and Ruben, and my sisters, Aurora and Paulina, for their unconditional love and support, and for their constant motivation and inspiration throughout my life.

I would also like to extend my appreciation towards my friends and the rest of my family from Mexico, for showing me care and support from afar during my year abroad.

Finally, I would like to thank Zain, a fellow Msc. student from the School of Informatics, for his sincere friendship and constant encouragement during my postgraduate studies.

Table of Contents

1 Introduction **1**

1.1 Contributions 2

1.2 Novelty 2

1.3 Overview of the Dissertation 3

2 Background **4**

2.1 Air Pollution 4

2.2 AirSpeck Sensors & the INHALE Project 5

2.3 Optimization Problems 5

2.3.1 Pathfinding 5

2.3.2 Orienteering Problem 6

2.4 Related Work 7

2.4.1 Sensing Spaces 7

2.4.2 Other research and applications 8

2.5 Software Tools 9

2.5.1 Open Street Maps 9

2.5.2 OpenStreetMap NetworkX (OSMnx) 10

2.5.3 Android Studio 10

2.5.4 Google APIs 10

2.5.5 Google Cloud Platform (GCP) 11

2.5.6 Flask & REST API protocols 11

3 Data Processing & Methodology **12**

3.1 Scope Definition 12

3.2 Air Pollution Data 13

3.3 Target Area 13

3.4 Road Network & Graph preprocessing 15

3.4.1 Resulting Graphs 17
3.5 Assigning Graph Weights 17

4 Algorithms **19**
4.0.1 Dijkstra's Algorithm 20
4.0.2 2-Phased Best Nearest Neighbor 22

5 Mobile App Development **24**
5.1 Mobile App Functioning 24
5.2 Mobile App Interface 25
5.3 Web server 26

6 Experiments, Result & Analysis **28**
6.1 Weighted Dijkstra, role of β 28
6.2 Second-Phased Best Nearest Neighbour 31
6.3 Real Trips Analysis 32
6.4 Mobile App Interface & Routes 36
6.4.1 Mobile App Metrics 38

7 Conclusions & Recommendations for Future Work **39**

Bibliography **41**

Chapter 1

Introduction

The escalating prevalence of pollutants in the atmosphere has drawn significant attention due to its wide-ranging and severe consequences on human health. In this regard, the steady upsurge of pollutants is correlated with an increased risk of cardiopulmonary and cerebrovascular diseases, a discernible decline in the quality of life, and a reduced lifespan [2][34]. Urban environments, being the epicenters of economic, industrial, transportation, and social dynamism, are especially vulnerable. These cities inadvertently become significant contributors to the proliferation of air pollutants [18], putting their inhabitants at heightened risk.

Given this scenario, the challenge of mitigating urban air pollution becomes a pressing concern that spans the domains of public health, urban planning, and sustainability. Cities, with their complex infrastructural setups and constantly changing human dynamics, face unique obstacles in lowering pollution levels. Protecting the health and well-being of urban populations require not only technological innovations but also a profound cultural shift.

On this subject, while there are web and mobile applications to traverse roads within cities, there is a gap in platforms offering to compute routes taking into consideration pollutant exposure, thus leaving an important source of health affectation disregarded. This project focuses on this gap by offering cleaner routes within an urban centre, in particular London's central area, providing a valuable offer for people interested in reducing their exposure to dangerous air pollutants

The project is positioned under a larger research group known as "Sensing Spaces" from the Centre for Speckled Computing at the University of Edinburg, and its main purpose is the design and development of a mobile application that computes and visualizes the routes with the least exposure to $PM_{2.5}$ pollutants, focusing on solving

two main optimization problems, namely pathfinding and the arc orienteering problem.

In this regard, this project works on two main questions: (1) Which is the route with the least exposure to airborne pollutants? Followed by (2) Which is the route with a predefined Target Distance (D_t) with the least exposure to airborne pollutants? [27]

While these questions are closely related to each other, they differ in their solution and overall problem category. The first focuses on going from point A to point B, and the second has an additional budget constraint or target distance D_t.

The scope of the project for route generation includes 2 modes of transportation, namely, walking and cycling. Additionally, it focuses on integrating and testing algorithms or metaheuristic approaches for route generation only, and leaves out of its scope the temporal spatial prediction of the pollutants.

1.1 Contributions

Throughout the duration of this project, a series of significant contributions were performed. Among these was the conversion of Open Street Map data into .csv formats, a critical step that enabled graph processing in both Java and Python platforms, focused on cycling and walking networks alike.

Additionally, in a bid to offer users a balanced route traversing experience, the author proposed a novel intermediate route. This new route type manages the extremes of the shortest and the cleanest paths, presenting users with a middle-ground solution that leverages distance and pollution.

On the application front, the Android Application was developed from the ground up. This application, allows users to generate and visualize routes. Concurrently, a REST API Flask application was built, designed as a prototype for the efficient transmission of pollution data.

Rounding off the project, the project was moved to the evaluation phase. Here, the proposed algorithms were put to the test, and an approach with routes generated from modelled data being was compared against real-world trip data, collected during July and August 2023.

1.2 Novelty

Firstly, on the route generation side, this project proposes a novel and convenient option for users when there are significant differences in length between the cleanest and

shortest route, as the cleanest route may prove unpractical in different scenarios. The third route proposed balances pollution and distance traversed, providing a reduction in pollution against the shortest route, and a reduction in length against the cleanest route.

Secondly, on the mobile application side, the application offers the distinctive feature to show the pollution levels on the screen, and not only the routes traversed. In this manner, offering valuable information to the users regarding both the routes generated, and also on the potential pollutant exposure levels on a given hour.

Thirdly, the methodologies and experiments introduced for comparing real-world and modeled routes help highlight the complexities of identifying the cleanest possible routes in practical scenarios. In this sense, these comparisons open doors for further studies into the challenges of route planning considering airborne pollutants exposure.

1.3 Overview of the Dissertation

The following chapter of this work will cover the relevant background of the problem focused on this work, including air pollution, data sources, related literature and main programming tools. In Chapter 2, the report will cover the data sources and preprocessing steps followed for its usage. Chapter 3 will focus on the algorithms used for solving pathfinding and arc-orienteering problems. Chapter 4, describes how the mobile application and REST API service developed work. In Chapter 5, the results of the algorithms proposed will be discussed, a route comparison will be performed from real walking trips with sensor data against modelled trips with predicted pollution data, and finally, an overview of the app interface and routes displayed there will follow. At the end of the report, in Chapter 6, the overall findings and limitations will be summarized and future recommendations will be suggested.

Chapter 2

Background

2.1 Air Pollution

Air pollution has its origins both from natural and anthropogenic activities, and yet, since the advent of the industrialization process in the 19th century, the share of the latter has increased drastically, posing a threat to human health [34] [27].

In this regard, airborne pollutants include Particulate Matter or PM (measured as $PM_{10}, PM_{2.5}$ and PM_1, according to the size in micrometres in diameter of the droplets that conform them), ammonia (NH_3), carbon monoxide (CO), nitrogen dioxide (NO_2), sulfur dioxide (SO_2) and ozone (O_3), among others. While all of these pollutants are formed through combustion (including fossil fuel and biomass combustion), transportation, and through industry and power generation-related activities, PM, in specific, originates in the atmosphere through chemical processes producing sulfates and nitrates and consists of tiny solid particles and liquid droplets, composed of acids, metals, dust, and soil.

The effects on human health of particulate matter, and $PM_{2.5}$ in particular, have been amply studied in the past years, and they have been linked with asthma, respiratory inflammation, impaired lung functions and even cancers [35]. Considering these hazardous particles, the World Health Organization has defined guidelines or limits for health protection in relation to air pollutants, in this sense, the annual and the 24-hour limits for $PM_{2.5}$ are 10 and 15 $\mu g/m^3$, respectively, and, for the case of PM_{10}, the annual and 24-hour limits are 15 and 45 $\mu g/m^3$ [34].

Following these current and past WHO recommendations, the United Kingdom and the city of London, in particular, have implemented various strategies for monitoring air contamination. For this purpose, the United Kingdom established the Automatic

Urban and Rural Network (AURN) monitoring network in 1972, and the city of London developed its own London Air Quality Network in 1993 [19] [31].

Along with these monitoring networks, some other private and academic projects have arisen to contribute towards air pollution measurement and forecasting in UK's capital city.

2.2 AirSpeck Sensors & the INHALE Project

The AirSpeck Sensors are a family of air quality monitors developed by the Centre for Speckled Computing from The University of Edinburgh [1]. They are divided into two models, the stationary AirsSpeck-S and the personal wearable AirSPeck-P. The AirSpeck-S is designed to be attached to street furniture such as posts or lamps, while the AirSpeck-P is a personal monitor designed to be used over clothing by pedestrians or cyclists.

Both sensors measure the concentration of $PM_{10}, PM_{2.5}, PM_1$ particles, as well as temperature and humidity in the ambient air, and are accompanied by a GPS sensor. The AirSpeck-S has a sampling rate of 5 minutes, whereas the AirSpeck-P has a sampling power of up to 5 seconds [37].

Both of these monitoring sensors have been used in the Inhale Program (Health Assessment Across Biological length scales for personal pollution exposure and its Mitigation) developed by Imperial College London in collaboration with The University of Edinburgh, to retrieve and assess air pollutant information and gather insights on their impacts on health, all this within the London area [16].

2.3 Optimization Problems

This section introduces the main optimization problems focused on in this work, the pathfinding and arc orienteering problems, along with a short overview of the algorithms or metaheuristics used to solve them, and their variants.

2.3.1 Pathfinding

The **Pathfinding** problem is a common optimization problem, which intends to identify the best path between two points in a graph according to predefined criteria or problems to solve (ex. find the shortest route, fastest route, etc).

In the simplest version, the goal is to find the shortest route between a source and target points, or nodes in a graph. For this purpose, a directed graph $G = (V,E)$ is traversed. For the graph G, a set of nodes $V = \{v_1, v_2, ..., v_n\}$ and a set of edges $E = \{(v_1, v_2), (v_2, v_3), ..., (v_{n-1}, v_n)\}$ are associated, each edge has a source node v_{src} pointing towards a target node v_{tar} and has associated weights $w(v_{src}, v_{tar})$ characterizing the feature associated to the edge [9].

In the classical formulation of the problem, the weights are related to distance and the goal is to find the route with the minimal sum of weights or distances. For the focus of this work, the weights are related to the $PM_{2.5}$ concentrations associated with the graph's edges.

Some of the feasible algorithms for pathfinding include A* or Bidirected A*, D and D-Lite, which focus on finding the route with the less cost associated with its traverse [9]. This work focuses on Dijkstra's Algorithm, invented in 1956 by the Dutch mathematician Edsger Dijkstra [10]. This algorithm was chosen given that previous work from the Sensing Spaces Team has shown that it performs well in the least polluted route problem [29] [12].

Finally, there is an additional version of the pathfinding problem when two different objectives are intended to be minimized. In this manner, the problem turns into a **multi-objective pathfinding problem**, which is an NP-Hard problem. It has been explored in applications that range from transportation of dangerous materials considering distance and risk or railway routes that consider transportation cost and time [5], [6].

Among the most common heuristics for this formulation of the problem is dynamic programming, like label correcting and label setting, that find solutions iteratively until finding a Pareto optimal solution [11]; bio-inspired algorithms such Ant-Colony which, mimicking the behaviour of the tiny insects on their quest of finding the shortest path to food, uses virtual ants to explore solutions by balancing exploration and exploitation, and in this way find the optimal or near-optimal solutions over various iterations [13]. Another method refers to variations on Dijkstra's Algorithm using similarity scores, or a weighted sum of weights [26] [15]. The latter is the method explored in this work and will be discussed in greater detail in later chapters.

2.3.2 Orienteering Problem

The **Orienteering problem** (OP) is a problem in the realm of combinatorial optimization, related to another famous Graph Theory problem, the Traveling Salesman Problem

(TSP). Yet, while the TSP focuses on visiting all the given points in the shortest possible route, the Orienteering problem introduces the concept of collecting scores or profits at each point, with a limited amount of available time [32].

In this problem, the goal is to determine the path starting from the start node v_{src} and ending at the destination point v_{tar} that the sum of the collected scores from visited nodes is maximized, and the total time (or distance or cost) of the path does not exceed the given budget.

The **Arc Orienteering Problem** (AOP), is another version of the OP, in which the player is required to traverse only through certain arcs or paths, instead of visiting nodes [33]. Thus this is the version of the problem-focused in this work, where the goal is to find the minimal pollution sum while traversing streets (edges), and the associated restriction or budget is the distance traversed (D_t).

Since the AOP is also an NP-Hard optimization problem, different closed-solutions and meta-heuristics have been proposed to solve it, including the Branch-and-Cut Algorithm [23], Iterated Local Search [33], or different modifications of the Ant Colony Algorithm [14].

2.4 Related Work

In this section, it will be discussed the related literature regarding pathfinding and arc orienteering projects revolving around air pollution. The first subsection covers current and previous internal projects developed by members of the Sensing Spaces Team, while the second is a broader literature overview.

2.4.1 Sensing Spaces

The idea of finding the least polluted route, within the Sensing Spaces team, was first explored by Ivanov in 2018. In his work he focused on finding the least polluted route for pedestrians, considering PM_{10}, in the city of Edinburgh. He explored methods for map data extraction, and various algorithms, including Dijkstra, A* and its bidirectional variants [29]. Yet, some of the challenges he faced were working with large areas.

Following Ivanov's project, in 2019 Sun further worked on generating the cleanest routes, using Dijkstra and A*, but this time focusing on a larger map within Edinburgh [12]. His work was based on walking networks as well, and, additionally, he developed a mobile application with an API REST service application to run the algorithms.

Concurrently, in 2019, Kumpalanuwat developed a similar project within Central Edinburgh focused on the AOP for biking networks, where he explored methods for processing graphs, implemented two metaheuristics, 2 Phased Best Nearest Network and Iterative Local Search and compared them against the branch-and-cut closed solution [28]. Both Kumpalanuwat and Sun focused their projects on $PM_{2.5}$ concentration levels.

Finally, during the same period of time that this work was undertaken, Welsh worked on ”Spatial Estimation of PM2.5 Using Transfer Learning” [17]. This related work focuses on the prediction of $PM_{2.5}$ pollutants in the cities of London in the United Kingdom, Delhi in India and Leon and Guadalajara in Mexico. Given the input of sensors from public local and AirSpeck sensors, additioned with weather and geographical features, the concentration of $PM_{2.5}$ is inferred in London city, and the predictions obtained are then converted into a pollution grid.

2.4.2 Other research and applications

The focus on finding the optimal route with regard to air quality has seen growing interest in recent times, following the impacts of air pollution on human health. In this regard, multiple work has been done using Dijkstra’s algorithm, yet the methodology and air pollutants vary, as well as the format to present the routes. For example, Deep proposes the usage of Dijkstra by using as pollution an Air Quality Index from public monitoring data in Dehli, but without a user interface [8]. In a different fashion, Zou et al. developed a Land Use Regression (LUR) model for pollutants prediction, with Dijkstra’s algorithm to generate the ”healthiest” route in the city of Beijing, using a mobile application connected to a web server that handles the processing [38].

A similar approach to the one proposed in this work is Green Path, which uses Dijkstra’s algorithm to find the optimal route for pedestrians and cyclists in the city of Helsinki [3]. In this project, users can specify the type of route they desire to follow according to the one with the most greenery or the cleanest air route, where air pollution was defined with a composite measure of NO_2, $PM_{2.5}$, PM_{10}, Ozone (O_3) and Sulphur dioxide (SO2), and the visualization is done through a web page.

Another similar project that effectively run from 2016 to 2019 was Clean Air Walking Routes developed by Cross River Partnership, where users were allowed to select an A-to-B reduced pollution route considering NO_2, $PM_{2.5}$, PM_{10} in the city of London, based on LAQN sensor data and presented through a web application, yet it is not specified which method for generating the routes it was used [7].

Finally, developed Cyclevancouver, a web service for finding the best cycling routes according to different criteria selected by the user, including the option to select paths with the least pollution exposure based on mean NO2 levels, but expanded to add different road characteristics such as elevation or greenery, as well [30].

In this regard, most of the previous work focused on finding A-to-B routes, associated to the AOP problem, there has been developed work specializing in finding routes with different criteria, including, once again, routes with the most green scenery and tourist attractions, routes with the best road conditions traversing urban and rural areas [20] [33] .

2.5 Software Tools

This section provides an overview of the main software, programming tools and web services used throughout the project.

2.5.1 Open Street Maps

Open Street Maps is an online open-source mapping platform for geographical data sharing and editing [24]. The map data from OpenStreetMap is stored in a structured format, consisting of nodes (representing points on the map) and edges (representing linear features like streets or roads).

Each element on the OSM map has associated with certain attributes, nodes contain a unique ID, followed by its latitude and longitude position, and the edges provide street information such as length, road type (walkable, driveway, etc.), name street, max. speed for driving, etc. For the interest of this project, only information regarding the geographical position of the nodes and lengths of the edges was considered from OSM.

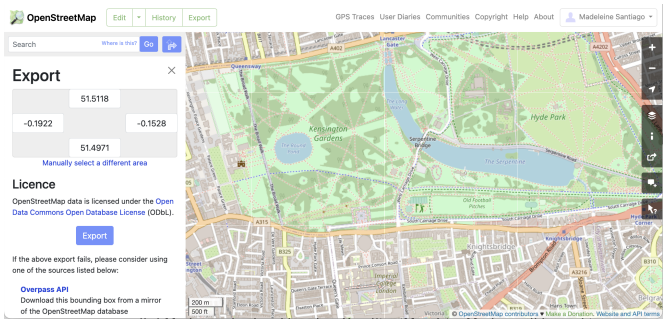

Figure 2.1: OpenStreetMap.org web page

2.5.2 OpenStreetMap NetworkX (OSMnx)

OpenStreetMap NetworkX (OSMnx) is a Python library that enables retrieving and processing of OpenStreetMaps data into a graph-based format [4].

Through the usage of this library, the map information about the underlying network structure of our area of interest, the street segments' and the topology was obtained and processed to create the files with the graph data (the nodes and the edges, as well as their attributes).

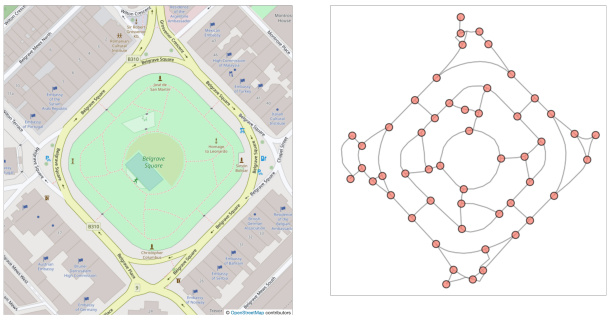

Figure 2.2: Example of OpenStreetMaps area processed with OSMNX and converted into a graph format, where the salmon dots represent nodes (position in space) and the connecting lines are edges (street segments).

2.5.3 Android Studio

Android Studio is an integrated development environment (IDE) owned by Google and designed for developing Android applications. It provides a large set of tools that allow developers in building Android apps, both from the backend or operations side as well as from the frontend or user interaction[36]. This work was developed using Android Studio Flamingo (2022.2.1 Patch), using Java as the main backend programming language and featuring additional libraries like GraphJT [22] for graph and Volley for HTTPS requests handling.

2.5.4 Google APIs

Google allows free usage services of valuable functionalities such as maps visualisation with their Maps API and geographical and location information such as museums, restaurants or places of interest through Places API. It is relevant to mention that Open Street Maps provides map and graph information yet, it does not support a native

environment for Android Applications, and conversely, Google provides a visualisation framework that can be seamlessly added to an Android mobile app, but not the connections or street information required for the project. Therefore the map data was obtained from OSM and the visualisation and places of interest information was done through Google Maps and Places APIs.

2.5.5 Google Cloud Platform (GCP)

Just as Android Studio provides an integrated development environment for developing Android Applications, Google Cloud Platform provides a comprehensive toolset for developing web and cloud-based applications. For the purposes of the project, the algorithm implementation and visualization were developed within the mobile application, but the possibility to add dynamic data was done on the online web server side using the Google App Engine service.

2.5.6 Flask & REST API protocols

A Representational State Transfer or REST API refer to a set of conventions and protocols for creating, reading, updating, and deleting resources over the web. Flask is a lightweight web framework developed in Python, it provides a flexible framework to develop dynamic web applications, that can be accompanied with a REST protocol allowing to send information over HTTPS requests (POST, GET, UPDATE, etc.). [25].

Chapter 3

Data Processing & Methodology

3.1 Scope Definition

The development of this work was focused on finding routes within the catchment area of the Inhale project. Therefore, the graph processing and routes tested in this work were based in the geographical area of central London, especially in the boroughs of Hammersmith & Fulham and Kensington & Chelsea.

This area encompasses a diverse range of land uses such as residential areas, semi-industrial zones, parks, and other green spaces, and is ruled by the main traversing or living areas of the Inhale participants but serves well for other targeted users of the application.

Additionally, the underlying street data and topology are based on the available OpenStreetMaps data as of August 2023, where all public paths for cycling and walking were included. In a later section of this chapter, it will be explained in greater detail the area covered and the networks used.

The project does not cover in its scope the pollution prediction or development of pollution maps. In this regard, the pollution prediction was obtained using Welsh's work [17], using as basis data obtained from varied public air pollution surveillance networks, LAQN and the AURN, and from the AirSpeck sensors deployed by the Centre for Speckled Computing for the Inhale Project. Finally, as pollution data, only $PM_{2.5}$ information is considered, leaving other pollutants like PM_{10}, PM_1 and gases, nitrogen dioxide and ozone out.

The methods for data collection and preprocessing are discussed next.

3.2 Air Pollution Data

The Air Pollution data used in this project was obtained from publicly available air pollution sensors from the United Kingdom's air monitoring networks, specifically the LAQN and AURN sensors deployed in the city of London, and through the AirSpeck sensors, deployed by the University of Edinburgh in conjunction with the Imperial College London for the Inhale project.

It is important to mention that pollution levels both measured by these sensors and the predicted values used as input for this project are not static and change throughout the day, and for this reason, the routes generated are not fixed either, nor the final sum of pollutants found with the algorithms proposed.

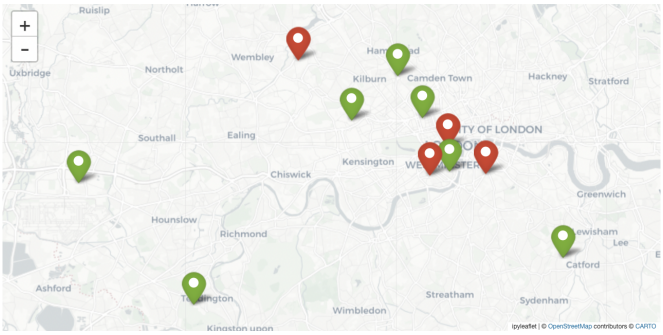

Figure 3.1: Public pollution monitoring sensors used for the data prediction stage (green: AURN, red: LAQN networks).

3.3 Target Area

A decisive point in the development of the project was the definition of the target area. This had different design considerations and implications. Firstly, while a larger covered area would be desirable, it would however result in a larger graph with an increased number of nodes and edges. This represents a memory constraint for the application that handles the system files.

Secondly, a larger area requires a larger number of tiles to maintain the spatial resolution. Which, has a direct limitation: the amount of data sent through the REST API request for updates.

As mentioned previously, the objective of the project was the development of a mobile application targeted to the users of the Inhale project. Therefore, the first step for the creation of the graphs and the covered area for the project was identifying the

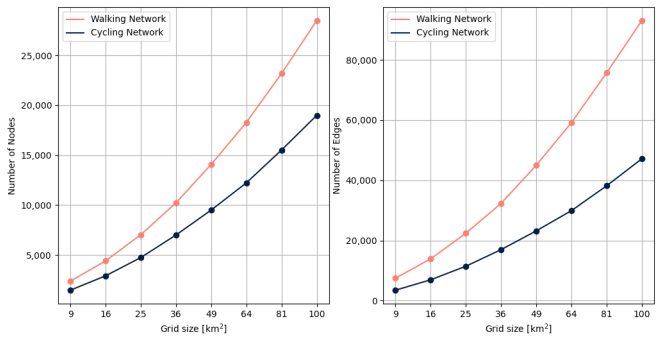

Figure 3.2: Graph Size Comparison by area covered in km^2.

traversing area of the project users. For this purpose, historical data from 2021 to 2023 from 137 AirSpeck personal sensors associated with different users of the project was processed. This allowed the proper identification of the best area to cover.

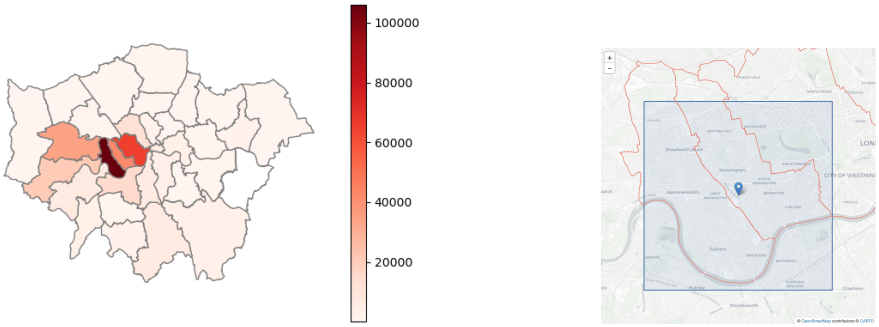

Figure 3.3: (Left) Number of unique samples from AirSpeck personal sensors per borough and (Right) Area Covered in Central London (shaded blue square).

From Figure 3.3, it can be observed that the boroughs with the highest incidence points were Hammersmith & Fulham, Kensington & Chelsea, and Westminster, with lower incidence in the Wandsworth and Ealing boroughs.

Taking into consideration the catchment area of the users, and the engineering constraints aforementioned, the area of interest lies within the square defined by covered lies within the geographical points of (51.520N, 0.149W) and (51.461N, 0.243W), equivalent to an area of 6.5 x 6.5 km^2 area in central London, tessellated into 3600 tiles, each of size of 108 x 108 m.

3.4 Road Network & Graph preprocessing

Having defined our area of interest, the next step followed was obtaining and processing the OSM data. For this purpose, the OSM data was extracted with the OSMnx Python library. Here is important to make some annotations on the network obtained and its components as a graph:

- **Nodes**: they represent single geographic points with specific latitude and longitude coordinates, each associated with a unique identifier or ID.
- **Edges**: also referred to as ways, are ordered lists of nodes that represent street segments.

Some sections of the network are more densely covered by nodes, for example, roundabouts or intersections, and naturally, the arcs or edges that connect these nodes are shorter. Some other sections of the network, the ones that represent large tunnels or bridges are less densely packed with nodes as they have fewer connections to other streets.

It is relevant to consider that, while the traversing mode of the project covers the same geographical area, the routes or networks differ between them. This is to say that not all streets can be traversed equally by bike or by foot.

In particular, the walking network in London is considerably larger than the biking one. What is more, all walkable streets are bidirectional while some streets are only unidirectional for cycling lanes. This proved to be an important processing and design consideration.

Figure 3.4: Example of the road network G (Left) and G_b (Right) for Walking and Cycling, respectively. In orange, the unidirectional edges for the Cycling network.

Firstly, the distinction between cycle and walkable networks and their features required the creation and processing of two different graphs: an undirected graph for

walking G (completely two-way directed edges) and a multi-directed graph for cycling G_b, where both graphs allow parallel connections between nodes (as parallel streets may connect to the same nodes), but not self-loops.

Secondly, the task at hand also required different processing for the walking and cycling graphs obtained. That is, the arc orienteering and the pathfinding problem have different requirements for its best implementation.

In this regard, the arc orienteering or circuit creation problem benefits from a graph with little or no dead ends since traversing multiple times or repeating the same segments in a circuit may provide a less satisfying experience than a circuit with rare segment repetition [28]. The pathfinding problem, on the other side, may require finding the path to a destination set on a street with no further connections, therefore a graph with the largest covered street network is best.

Figure 3.5: Example of a walking road network graph G before and after the dead-ends were removed.

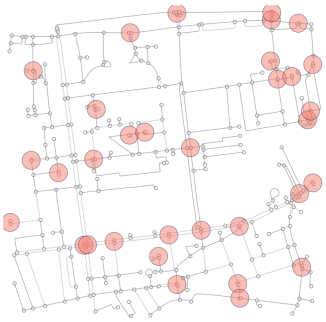

Figure 3.6: Example of a cycling road network graph G_b where nodes that were too close together have been merged into a single one in the cluster's centroid.

Following this difference in requirements was dealt with by developing two more graphs, a set of "pruned" or with no dead ends or cul-de-sac graphs for walking and

biking (G_p and G_{bp}), in addition to the main graphs. The processing step resulted then in a total set of 4 graphs, with different numbers of nodes and edges.

Finally, one last step for preprocessing was undertaken: the consolidation or merging of nodes too close to each other. In this step, all nodes that were found within a distance smaller than 5 meters were merged into a single node. This was mostly a simplification step since the topology of the streets did not change.

3.4.1 Resulting Graphs

All this preprocessing resulted in 4 graphs with a distinct number of nodes and edges, as seen in Table 1. As expected, the smallest graph is the pruned graph for cycling, G_{bp}, and the largest is the complete or main graph for walking, G.

Graph	Number of Nodes	Number of Edges
Main Walking Network G	14134	40946
Pruned Walking Network G_p	10407	33946
Main Biking Network G_b	10123	23860
Pruned Biking Network G_{bp}	6479	16751

Table 3.1: Resulting 4 Graphs Information

3.5 Assigning Graph Weights

This section focuses on the logic followed for assigning the weights w on the graphs' set of edges E. This is a crucial element of the process since the algorithms implemented are designed to find the optimal route according to the definition and assignment of the weights. The two weights assigned to the graph's edges used throughout the project are:

- **Length**: The physical distance between nodes in a street, measured in meters. This was directly obtained through Open Street Maps.
- **Pollution**: $PM_{2.5}$ levels obtained from the pollution grid generated by Welsh [17], measured in $\mu g/m^3$. Each tile from the grid is characterized by a pollution value. In this case, the $PM_{2.5}$ values are assigned to the nodes of the graph located within the tiles and then the edges were assigned with the average value between source and target node. This is especially significant when the edges cover multiple tiles.

It is also relevant to mention that the weights assigned for pollution are not fixed, since the pollution levels and their distribution change throughout the day. In this regard, the mobile application and the data structures were developed with a weight update functionality in mind.

Figure 3.7: Example for July 20, 2023 at 8AM for the 60x60 pollution grid and the weights assigned for pollution for the graph G. The left figure shows the pollution grid and the right one the transformed grid values into the weights assigned for the graph's edges.

Chapter 4

Algorithms

This section will be focused on explaining the algorithms and heuristics for Pathfinding and Arc Orienteering problems considered in this project, as well as their baseline implementation. With this considered, the main algorithms are **Dijkstra's** and **2 Phased Best Nearest Neighbour**.

To provide a comprehensive understanding of the steps involved in executing the algorithms, Figure 4.1 presents a high-level overview.

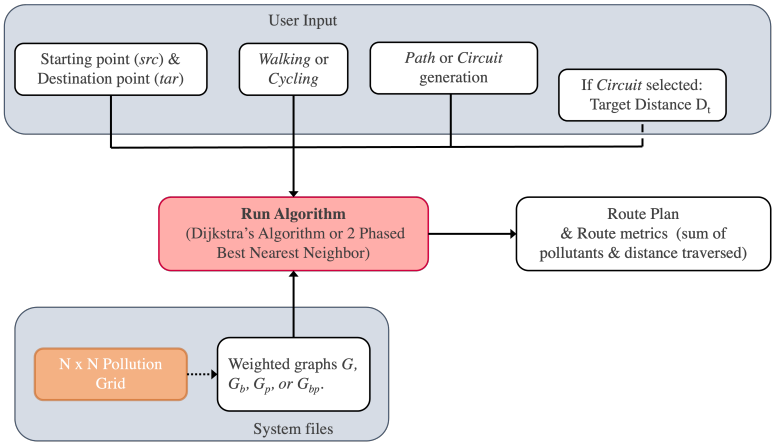

Figure 4.1: Process overview for running the algorithms.

Before proceeding to the Algorithms, one element of the implementation must be discussed. The appropriate method to measure the distance between point A and point B in a spherical surface, like the Earth's, is through the haversine distance and not the Euclidean distance [21].

For this, given two points on a sphere with radii r, latitude ϕ_1 and ϕ_2, and longitude λ_1 and λ_2:

$$a = \sin^2\left(\frac{\Delta\phi}{2}\right) + \cos(\phi_1)\cdot\cos(\phi_2)\cdot\sin^2\left(\frac{\Delta\lambda}{2}\right)$$
$$c = 2\cdot\text{atan2}\left(\sqrt{a}, \sqrt{1-a}\right)$$
$$h = r\cdot c$$

Where $\Delta\phi = \phi_2 - \phi_1$, $\Delta\lambda = \lambda_2 - \lambda_1$, and h refers to the haversine distance.

4.0.1 Dijkstra's Algorithm

Dijkstra's Algorithm is one of the most common pathfinding algorithms, where the goal is to find the best route or path from a *src* node to a *tar* node, the algorithm traverses the graph and, if it exists, it outputs the route with the minimum sum of weight assigned to the edges traversed [10].

In Algorithm 1 the logic followed by Dijkstra's algorithm is profiled.

Algorithm 1 Dijkstra's Algorithm

```
1:  procedure DIJKSTRA(G, s)
2:      Q ← set of all nodes in G
3:      for all v ∈ Q do
4:          dist[v] ← ∞
5:          prev[v] ← undefined
6:      end for
7:      dist[s] ← 0
8:      while Q ≠ ∅ do
9:          u ← node with the smallest dist in Q
10:         remove u from Q
11:         for all neighbor ∈ neighbors(u) do
12:             alt ← dist[u] + length(u, neighbor)
13:             if alt < dist[neighbor] then
14:                 dist[neighbor] ← alt
15:                 prev[neighbor] ← u
16:             end if
17:         end for
18:     end while
19:     return dist, prev
20: end procedure
```

Despite its simplicity, Dijkstra's algorithm is a reliable algorithm used in multiple types of problems, since it provides flexibility in how the weights are assigned, although it does not allow for negative weights. As a first approach, it was tested on the shortest path and compared with online mapping sources.

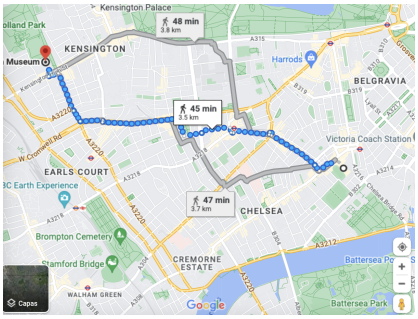

Figure 4.2: Google's shortest route.

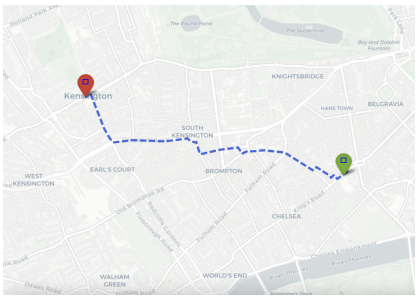

Figure 4.3: My shortest route for walking.

4.0.1.1 Weighted Dijkstra

Weighted Dijkstra is an extension of the regular algorithm applied to multi-optimization pathfinding problems. As the name suggests, the different feature values are normalized, weighted given a β factor, summed and then assigned to the edges [15].

The steps required for processing the weights are the following:

For each edge E and for each weight type w_n, a matrix A is obtained such that

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

Then each element is normalized by the sum of its values.

$$r_{uv} = \frac{a_{uv}}{\sum a_{uv}}$$

$$R = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ r_{m1} & r_{m2} & \cdots & r_{mn} \end{bmatrix}$$

Lastly, by the weighted sum method we obtain the new edges' weights:

$$c_{uv} = \sum W_v r_{uv}$$

$$X = \begin{bmatrix} w_1 r_{11} & w_2 r_{12} & \cdots & w_n r_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ w_1 r_{m1} & w_2 r_{m2} & \cdots & w_n r_{mn} \end{bmatrix}$$

Where X is now the matrix with the new weighted edges. At this point, the pathfinding problem can be solved with Dijkstra's algorithm normally.

The relevance of considering a multi-objective optimization problem for routing derives from the differences between the shortest and cleanest route, since in some scenarios, this difference may prove too large and thus become undesirable or unpractical for the users [12, 29].

By providing the option of a trade-off route users can still lower their air pollutants exposure, yet maintain a closer-to-shortest distance route. Additionally, the weighted Dijkstra algorithm also allows for different ratios in weight, thus allowing experimentation regarding the importance of each feature, as seen in Chapter 6.

4.0.2 2-Phased Best Nearest Neighbor

In the case of the Arc orienteering problem, the algorithm implemented is a 2-Phased Best Nearest Neighbor, an adaptation of the Best Nearest Neighbor Algorithm proposed by Kumpalanuwat [28].

In the first phase, which can be defined as an exploratory one, the algorithm greedily searches for the best neighbour of its current node (the neighbouring node with optimal defined value) and moves towards it. It continues to do so until a reserve of distance, the approximate distance it requires to get to the target node is depleted. From this point on, in the second phase, the algorithm finds its way towards the target node using Dijkstra as a subroutine. In Algorithm 2, the process followed by the 2-Phased Algorithm is outlined.

On this subject, the reserve of distance or budget associated with this algorithm takes relevance with the value f, which is a preprocessed ratio with respect to each of the graphs (pruned graphs G_p and G_{b_p}). In this sense, it is defined as the ratio between the haversine and shortest route distance (by Dijkstra's) distance between *src* and *tar*.

In addition, this metaheuristic comes with two tunning stages, one associated with the penalties for revisiting edges, which refers to the inclination of the algorithm to traverse the same edges it has previously visited if they are again the best neighbours, and another one for determining the optimal f value. In this work, the penalty and f values used varied for both network Graphs G_p and G_{b_p}.

Algorithm 2 Best Nearest Neighbor

procedure BEST NEAREST NEIGHBOR(G, src, tar, D_t)
    $route \leftarrow [src]$
    $curr \leftarrow src$
    $D \leftarrow 0$
    while $D + (f * distance(curr, tar)) < D_t$ **do**
        $options \leftarrow list(out(curr))$
        $candidate \leftarrow min(cost(options))$
        $sumDist \leftarrow sumDist + dist(curr, candidate)$
        $curr \leftarrow candidate$
        add $candidate$ to $route$
    end while
    $path$ = Dijkstra(curr, tar, weight = ”pollution”)
    add $path$ to $route$
    return $route$
end procedure

Chapter 5

Mobile App Development

The implementation of the algorithms has been discussed in the previous section, this section introduces the development of the mobile application, the methods to implement the algorithms, the Mobile App interface and the preparation of a REST API web server in Google Cloud Platform.

5.1 Mobile App Functioning

The mobile application was developed considering handling multiple types of requests by the user, namely, finding routes between 2 different points and finding circuits between 2 different points and a target distance D_t.

With this in mind, the algorithms and metaheuristics developed for each type of request run locally within the application.

In order to handle the processed graphs, 3 Main objects were created:

1. **Object Tile**: Stores tiles coordinates as a polygon object and the pollution information, its ID and the current predicted $PM_{2.5}$ value.

2. **Object Node**: Stores the node information as seen from OSM. It has as attributes: ID, Latitude, Longitude, Tile ID it belongs to, and Pollution Value

3. **Object Edge**: Stores edges attributes, which are: Source and Target Nodes, Pollution and Length.

When the pollution grid is updated, each tile gets assigned a new value, which results in each node that is assigned to that tile getting updated as well. This is done upon initialization or at command with the $UpdatePollution$ button.

The actual graphs are then built by using Java's library for graphs GraphJT. As mentioned previously, we obtain 2 sets of multidirected and undirected graphs for cycling and walking networks, respectively.

In a general sense, the application runs 3 main programming blocks:

1. **Initialisation**:
    - Reads CSV files saved within the app resource folder, which contain the graphs' information, it then creates the graphs and related objects with their corresponding features.
    - Initialise Map View and Widgets (buttons, search bars, text views, etc.)
2. **Handle User requests**:
    - Loads appropriate content according to Route or Circuit: For example, for Circuit the menu shows a target D_t slider, too.
    - Show and update the user-selected Start and Ending Points.
    - Visualize pollution upon request.
    - Run Algorithms on command; (*runDijkstra* or *runBNN*), plot routes and their metrics (considering transportation method).
    - Handle errors: some main wrongdoings were identified with a poka-yoke approach, (i.e. requesting route without defining source or destination or not granting internet or location permission).
3. **Update weights**: connect to an HTTP server to load the changing pollution grids.

5.2 Mobile App Interface

The App Interface consists of only one Layout, which includes a map and a menu section. The map is dynamic and allows clicking on it for setting or modifying the destination. All of the information and functionalities, found at the bottom of the screen, include searching for a new destination or origin location, modifying the transport mode or choosing between finding a circuit or a direct path route.

In the instances in which a prohibited command is requested (i.e. route without source or destination) a warning box appears.

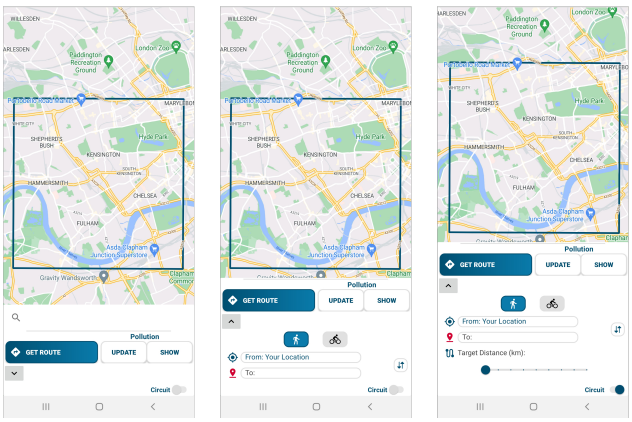

Figure 5.1: Mobile App Layout Overview.

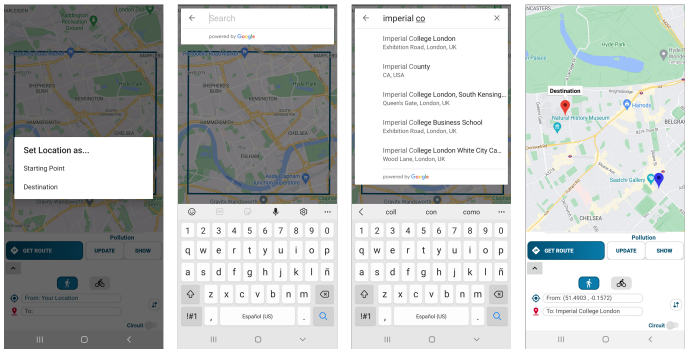

Figure 5.2: Mobile App start and destination selection options

Besides the main requests for a new route or circuit trip, an additional functionality of the application is the visualization of the pollution across the area of interest, upon user request. While it sounds simple, it was an operational challenge since the grid contains 3600 tiles, and loading them made the application fail due to overcapacity. The method to overcome this was by dividing the tiles to plot into batches, managing the load with a handler() object and waiting 500ms between batches. It is the most resource-consuming process of the application but it is a valuable feature, both for the user to verify the routes displayed and for simple interest in knowing which areas are the most polluted. Yet, for a user concerned with following the routes only, there is the option to hide the pollution grid, as well.

5.3 Web server

A Flask REST API application was developed using Google Cloud App Engine, to serve as a prototype for sending real-time pollution data. Due to time constraints on building the complete real-time-predictions pipeline and the fact that there exists, at the

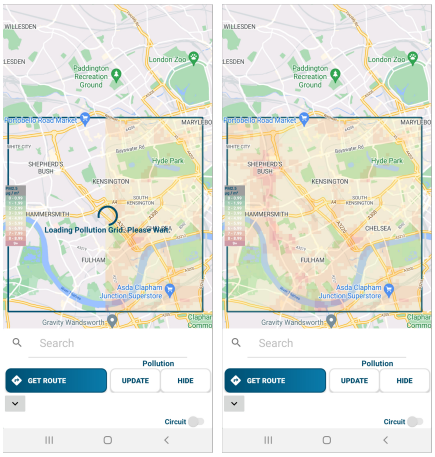

Figure 5.3: Mobile App View Pollution Function.

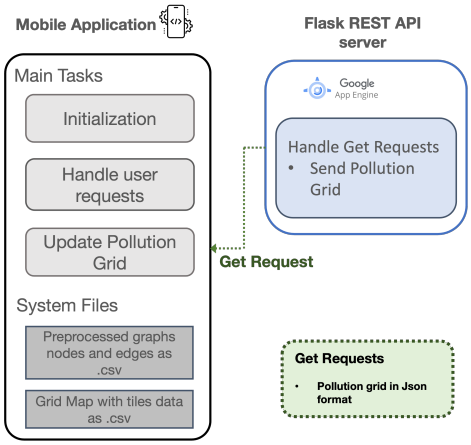

Figure 5.4: Mobile App View Pollution Function.

time of this work, a fixed 3-hour lag between the current hour and the data available on the LAQN and AURN Public Network web pages and used as input required to perform the predictions, the server contains only the REST API application.

In the current stage, the server contains the pollution prediction from one day only (July 20, 2023), which can be retrieved with the ”Update Pollution” button from the application. The data obtained from this request follows the JSON format of {data: [tileID$_1$: PM$_{2.5}$ value$_1$, tileID$_2$: PM$_{2.5}$ value$_2$, ... , tileID$_{3600}$: PM$_{2.5}$ value$_{3600}$]}. The tileID from the request message matches with the tileID from the grid, so the function *updateWeights* swiftly modifies the pollution grid within the app and is then followed by an update procedure for the graphs' weights.

Finally, the current configuration between the mobile app and the server can be visualized in Figure 5.4.

Chapter 6

Experiments, Result & Analysis

In this chapter, the main results and findings from the experiments will be visualized and discussed. The first set of experiments focus on evaluating the algorithms proposed and the different methods for assigning the weights to the graph edges. The second set of experiments relates to the performance of the routes developed by the algorithms and its comparison against real trips taken by users during July and August 2023, using predicted $PM_{2.5}$ data from the work of Welsh [17]. The last section of the chapter focuses on discussing the functionalities of the mobile app.

6.1 Weighted Dijkstra, role of β

How the weights are assigned to the edges is one of the essential tasks for pathfinding. Since, according to the weights and their definition of the problem, the ideal route will be found.

As mentioned in previous chapters, in this work, the least polluted route is considered the route that has the minimum cumulative sum of $PM_{2.5}$ pollutants. As Ivanov mentions, there is a relationship between the shortest and the cleanest route [29]. This is because, in real life, a longer route would naturally have a larger number of pollutants, he argues. Yet, both he and Sun also recall how the shortest and cleanest route does not necessarily the same, as there are areas within a city or a grid, that experience higher pollution levels [12].

In order to obtain a route that reconciles pollution exposure and distance, a weighted Dijkstra approach was explored. In this regard, there is a modifiable factor $\beta = \{0, 1\}$ that accounts for how much importance is given to one or the other feature (length or pollution), after a standardization of the variables. In the next few pages, it will be

discussed how changing this factor alters the ideal route found.

First, let us understand the metrics of the proposed basic routes. In this experiment, 3 sets of 100 source and destination searches were done. For each source and target points the routes for shortest, cleanest and routes with intermediate values using different values for β were generated.

The sets are divided by short (1000m $<$ S $\leq$ 1500m), medium (2500m $<$ M $\leq$ 5000m) and long (5000m $<$ L $\leq$ 7500m) routes. The distance used to define if it was a short or long route was the total length of the route between *tar* and *src* node using the shortest path found by Dijkstra's algorithm. The experiment was performed in the walking graph *G*, with the pollution prediction grid for 20 July 2023 at 7:00 AM.

In Figure 6.1 it can be observed how differently the cleanest and shortest routes compare both in total length and in total sum of pollutants, for the pollution grid and graph aforementioned. As expected, longer routes show a larger pollution sum.

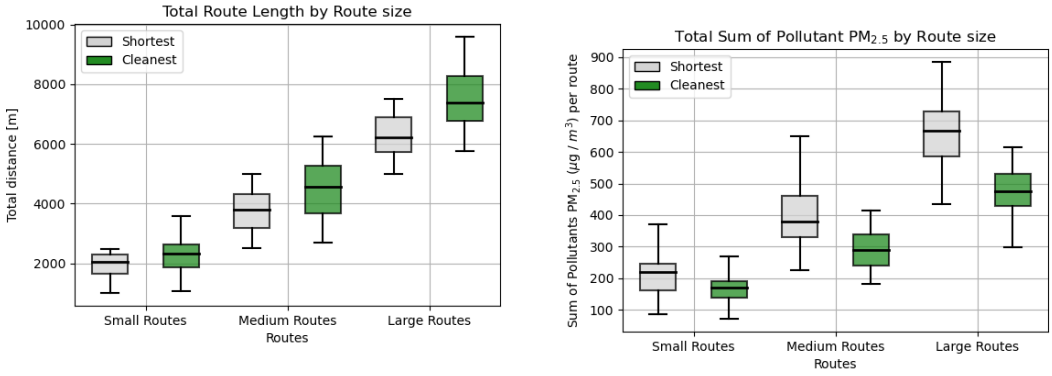

Figure 6.1: Total Length and Total Sum of Pollutants by Route Size.

Next, for that same source and destination, the weighted average route was generated. The following plots show the metrics associated with the compromise route with a β = 0.5. These routes are expected to be a middle point between the shortest and cleanest. It can be observed how for each type of route the route serves as a compromise between length and pollution.

From Figure 6.2, 6.3, 6.4 it is worth noticing how the new route proposed presents reductions in length with a trade-off in pollution, this is especially appreciable for medium and longer routes.

Finally, for those same source and target destinations of the experiment, the β factor with values of 0.25 and 0.75 was tested. In Figure 6.5, which shows the behaviour of β on the long routes, a trend is distinguishable in which the by lowering the β factor

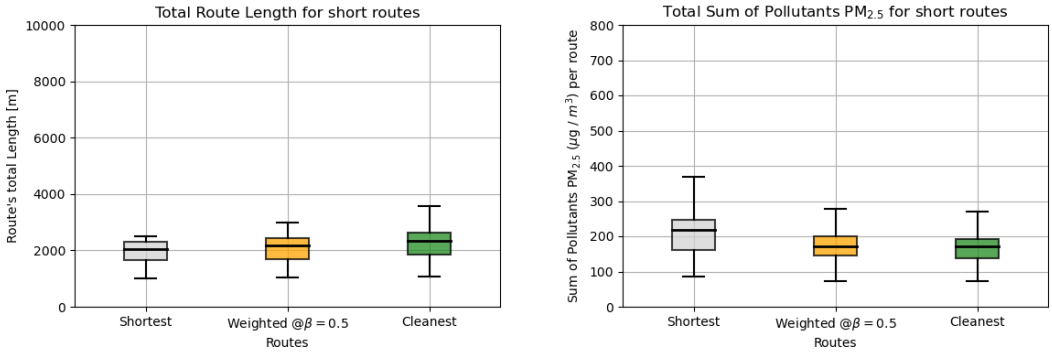

Figure 6.2: Behaviour of the routes with weighted Dijkstra with $\beta = 0.5$ for Total Length and Total Sum of Pollutants in Short Routes

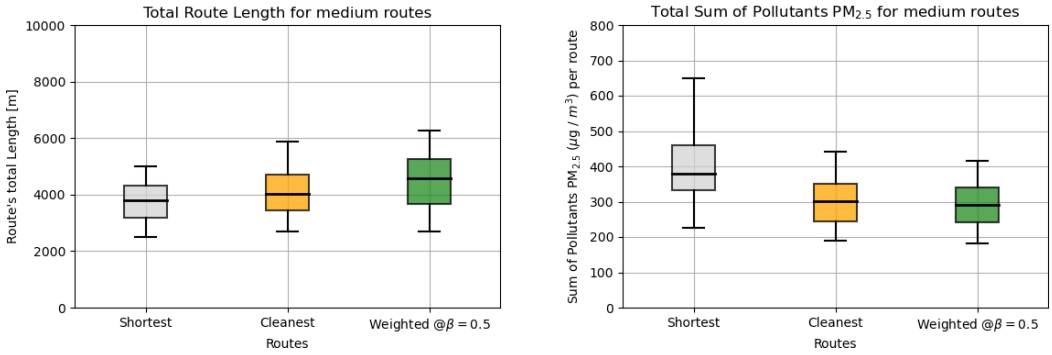

Figure 6.3: Behaviour of the routes with weighted Dijkstra with $\beta = 0.5$ for Total Length and Total Sum of Pollutants in Medium Length Routes

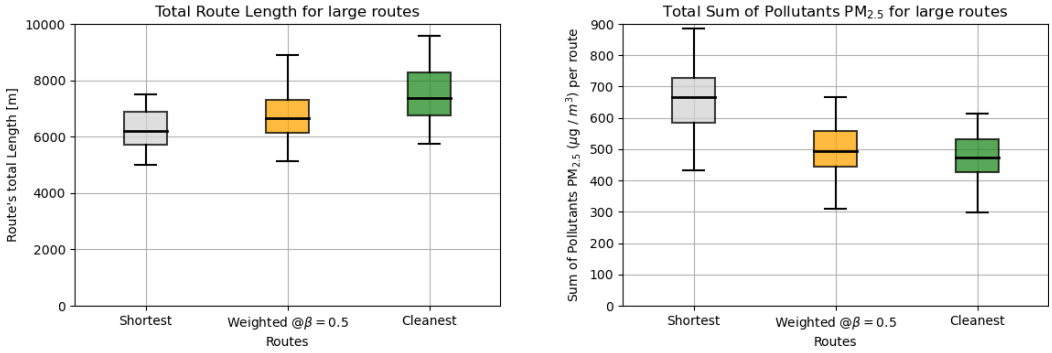

Figure 6.4: Behaviour of the routes with weighted Dijkstra with $\beta = 0.5$ for Total Length and Total Sum of Pollutants in Large Routes

the best routes found start decreasing in pollution value and augmenting in length, and conversely with a higher value for β.

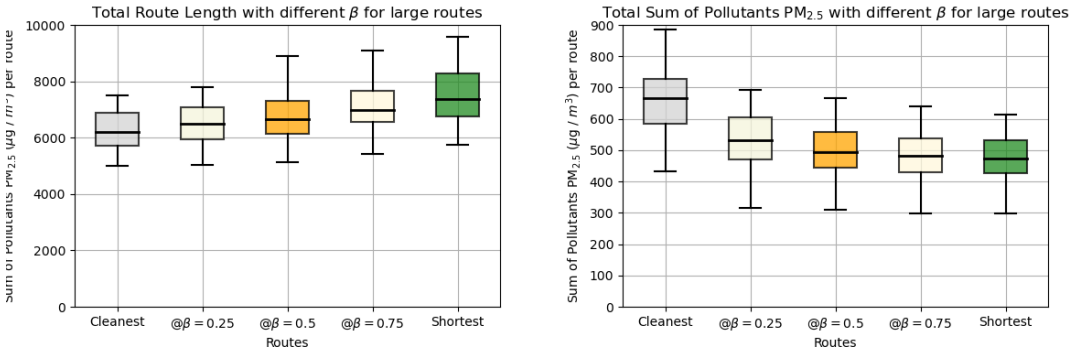

Figure 6.5: Influence of β for Length and Pollution when using weighted Dijkstra.

6.2 Second-Phased Best Nearest Neighbour

In this section, it will be discussed the performance of the Second-Phased Best Nearest Neighbour algorithm implemented for the arc-orienteering problem. In this case, the experiment consisted of generating a thousand different routes with the same source *src* and target *tar* nodes, thus creating a closed circuit, with different distance target D_t, from 1000 to 5000 meters in 1000 meters increments for both the walking and biking networks G_p and G_{b_p}, using the same pollution data as the previous experiment. The main goal of this experiment is to test how well the algorithm, after tunning, matches the target Distance D_t predefined.

Target Distance D_t	**Distance$_W$**	**Pollution$_W$**	**Distance$_B$**	**Pollution$_B$**
1000	1102 ± 155	75 ± 21	1129 ± 288	72 ± 27
2000	2126 ± 202	145 ± 34	2140 ± 345	137 ± 39
3000	3139 ± 203	213 ± 41	3165 ± 358	198 ± 48
4000	4168 ± 249	288 ± 50	4174 ± 377	257 ± 58
5000	5190 ± 277	359 ± 58	5185 ± 412	318 ± 68

Table 6.1: Walking and Cycling routes distance and pollution values for circuits with different D_t, where the pollution is the sum of pollutants measured in $\mu g/m^3$ found for graph G_p and G_{b_p} and distance is measured in meters.

It is relevant to mention that despite that the f value defined for the algorithm is regarding the ratio between the haversine and the distance measured by the shortest route, a tunning process was performed. In this case, the results shown next were done with the f value obtained with the distance measured by the medium route with $\beta = 0.5$, since the second phase is not done through the shortest route but with the cleanest route,

and as seen in the previous section, there is a considerable difference in length between one and the other. In this regard, using the f value computed between the haversine and the cleanest route distance might prove inefficient for changing pollution grids.

Lastly, the f values were computed separately for each graph, yet the penalty assigned for repeating edges was set to 2 in both cases, ensuring that the routes find the neighbouring edge with the least pollutant, while avoiding repeating the same edge multiple times.

Table 6.1 shows that while the algorithm implementation closely follows the target distance D_t demanded, for both cycling and walking networs, there is still an associated error and tendency to overcalculate the final distance.

6.3 Real Trips Analysis

In this section, it will be discussed the analysis of real walking trips taken during 5 different days in late July and early August 2023 (July 20th, 28th and 1st, 2nd and 3rd of August), from Ladbroke Grove to Imperial College London. The purpose of this experiment is to understand how the routes generated and the pollution metrics from the routes traversed compare against real data.

For all of the days, there are 2 or 3 different trips taken, which account for the trips from Ladbroke Grove to Imperial College, from Imperial College to Ladbroke Grove, or from Imperial College to a midpoint between the latter and Ladbroke Grove. With the exception of the trip from August 3, all trips from Ladbroke Grove to Imperial College avoid any route via Kensington Gardens, for all trips from Imperial College to Ladbroke Grove the route taken goes *through* Kensington Gardens. All trips were taken between 6:00 AM and 9:00 AM. In the case of the routes that traversed more than one fixed hour (for example from 7:55 AM to 8:30 AM), the pollution grid used for the modelled route was the fixed hour in which most of the route was taken (for example, 8 AM in the scenario exemplified before, as most of the route is taken within that hour).

To maintain in the following tables and figures Route 1 will refer to Ladbroke Grove to Imperial College, Route 2 will refer to Imperial College to the Midpoint, Route 3 will refer to this Midpoint to Ladbroke Grove, and finally, Route 4 will refer to the trip from Ladbroke Grove to Imperial College through the gardens, where Route 4b is the return trip following the same path.

The method for obtaining data was through the usage of an Airspeck Personal sensor, which retrieves data every 30-35 seconds, the data recorded and used in this experiment

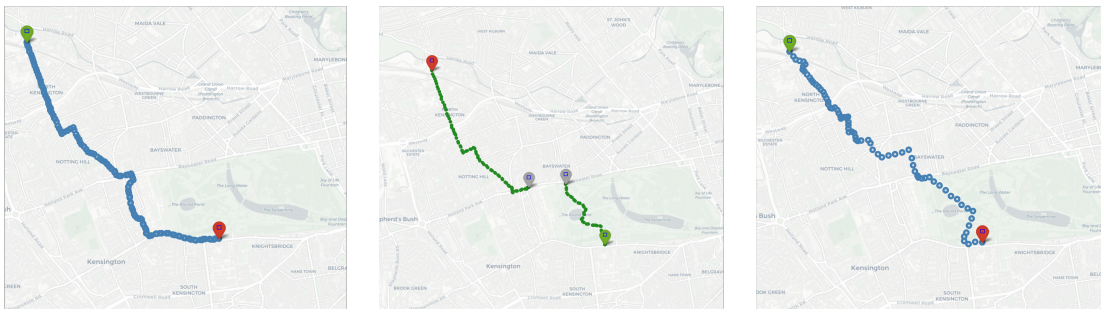

Figure 6.6: Routes taken for this experiment. The left image refers to Route 1, the centre image refers to the trip back with Route 2 (traversing through the park) and then Route 3, and finally, the right image refers to Route 4.

was Timestamp, Pollution [$PM_{2.5}$] levels, and GPS location (Longitude and Latitude). In order to compare the data obtained, through the GPS location samples, the real route taken was recreated by finding the closest nodes to the sensor location samples and running the shortest path running Dijkstra's algorithm, between samples, and then appending the new nodes and traversed edges to the path followed. The pollution grid used for the recreated route was obtained by running the prediction model [17] with the Airspeck, AURN, LAQN and weather data as seen at the hour of the day the real trip was taken, and the value assigned as $PM_{2.5}$ for the graph network was by following the regular logic of weight assignment for pollution.

It is important to consider two limitations of the experiment: The routes taken were recreated following the closest nodes to the GPS location samples; therefore to obtain the real trip distance the haversine distance between samples was used. For the modelled route the distance considered is retrieved directly from the edge object's length attribute, thus there are differences between the real distance traversed and the distances from the recreated route. The second limitation arises with the differences in pollution measured from the sensors and the one predicted in the pollution grid and assigned as the edge's pollution weight.

With this in mind, Figure 6.7 shows the recreation of one of the routes, specifically Route 4 from August 3rd 2023, and the pollution values obtained from one source to the other, ordered across the distance traversed while following the route. This route is specially relevant, since, from the model, it was repeatedly the least polluted route with the pollution grid maps from the days between 20 to 24 July between 6:00 and 9:00 AM.

As it can be seen from Figure 6.7 (right), there is a difficulty when trying to measure

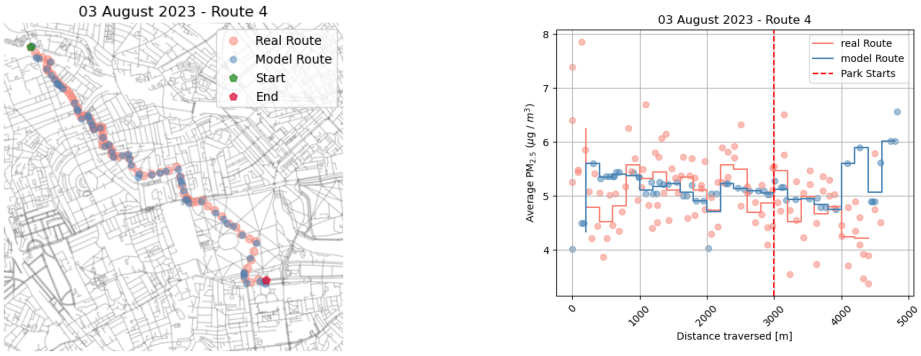

Figure 6.7: (Left) Real Route and Modelled Route for Route 4 (Ladbroke Grove to Imperial College via the park) and (Right) Pollutants range found across the route ordered by distance traversed

the total sum of pollution from one route to another, since the sensor would be time-based (samples in time) and the model-based route is not. A simple way of comparing one and the other route would be obtaining the sum of the total values observed with each route, yet there are more samples in the time-based route.

Trying to reduce the inconsistency between time samples, the route was segmented into 200 meters segments as it roughly relates to 3 minutes of walking, and the average pollution value, for each type of route, was obtained.

As it can be observed from Table 6.1, the values from real and model routes for distance remain within less than 10% difference from start to finish, so the sections traversed, and by extension, the pollution levels observed per section, are comparable.

From Table 6.1, it can be noticed that while the routes for the 3 types of trips are similar or nearly the same, the pollution values differ from one day to the other. In this regard is worthwhile to mention that the method for assigning and summing pollution values accounts for the changes in the pollution grid, as the values also change for the day and hour the route is being considered.

Despite the aforementioned expected and welcomed variations in pollution values, the values for similar trips in the modelled data remain in a consistent range, since the predicted pollution for each area does not change to such a strong extent as with real data.

Finally, from the segmentation of the route in distance and using the sum of the average values, it can be observed how the modelled route is regularly overpredicting the total sum of pollutants found within the route.

This behaviour can be explained by two reasons: firstly, there is an error associated

Date	Route	Distance$_R$	Distance$_R$	Pollution$_R$	Pollution$_M$
03/08/2023	Route 4	4.58	4.8	110.6	137.8
	Route 4b	4.58	4.87	100.5	124.1
02/08/2023	Route 1	4.60	4.73	96.2	117.8
01/08/2023	Route 1	4.52	4.46	179.7	121.6
	Route 2	1.25	1.35	39.8	40
28/07/2023	Route 1	4.22	4.32	26.9	103.3
	Route 2	1.35	1.35	4.2	27.6
	Route 3	3.07	3.19	22.9	63.6
20/07/2023	Route 1	4.52	4.69	52.3	107.4
	Route 2	1.33	1.36	11.8	26.7
	Route 3	2.71	2.7	33.5	72.5

Table 6.2: Real and modelled routes (Route$_R$ and Route$_M$) analyzed according to the total sum of averaged values $PM_{2.5}$, measured in $\mu g/m^3$ values every 200 meters.

with the pollution predictions. Secondly, the method proposed assigns weight to the edges of the graph solely on the pollution found between two nodes, so traversing one edge (which represents a street segment) can be seen as one sample. With the sensor data, there is a sample every 30 seconds, which of course means that during that time a person has traversed a certain distance that can cover less than an edge in the modelled data.

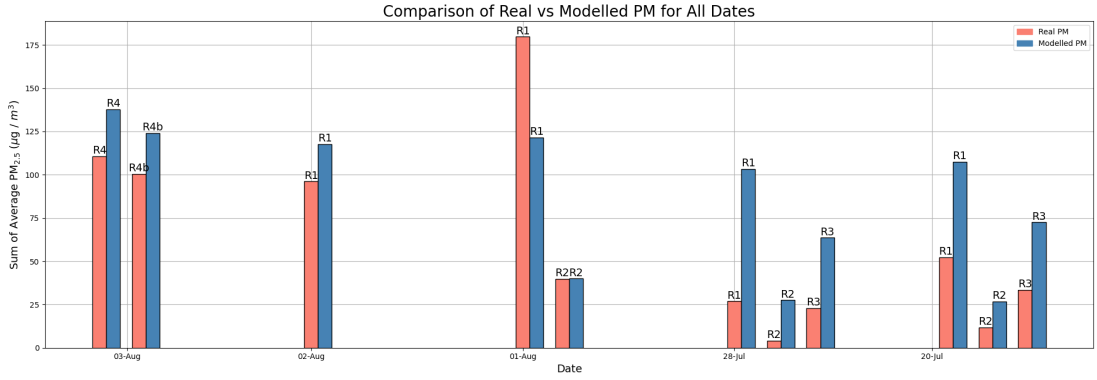

Figure 6.8: Comparison all Routes taken using the averaged sum of pollutants every 200 meters.

6.4 Mobile App Interface & Routes

This last section will be focused on a comparison of the routes generated in a single day (July 20, 2023), present how different routes or circuits are displayed on the mobile application, and provide some insight into the results. Next, a comparison of the metrics for the two routes will be discussed.

Hereof, Route A is from Ladbroke Grove to Imperial College London, and Route B is from the south of Fulham to the upper side of the borough (the exact location is The Climbing Hangar London to Ikea Hammersmith). In Figure 6.9 it can be appreciated Route A, while Route B is shown in Figure 6.10, along with the pollution evolution at 3 different times of the day [8 AM, 4 PM and 11 PM].

To keep it concise firstly, in Figure 6.9 it can be observed the mobile app handling of route generation by walking and cycling at 8 AM. It can be visualized how the routes differ according to the selected transportation method. In this regard, it is noticeable the behaviour of the medium route, and how it shows a reduction in the length as compared to the cleanest route while sacrificing pollution sum.

Here it is important to mention that the medium route is not displayed for the biking route because, in order to avoid overcrowding the interface, it is suggested only when the difference between the shortest and cleanest routes surpasses a 300 meters threshold and if the medium route is different to the other two routes.

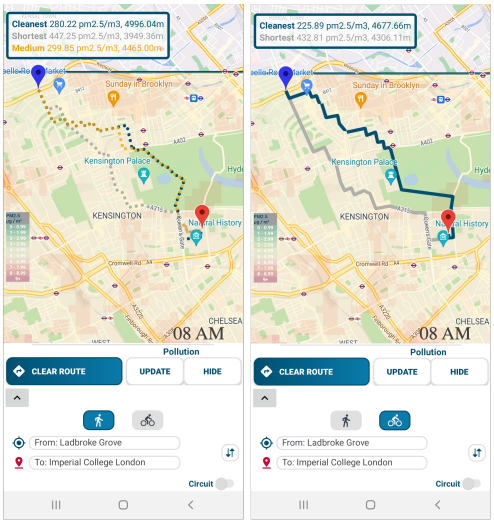

Figure 6.9: Example routes created by walking (right) and cycling (left) and associated metrics are shown in the App interface [Route A].

In Table 6.3 a comparison between the pollution and length metrics from different hours can be observed. An important aspect appears, despite the variation in pollution,

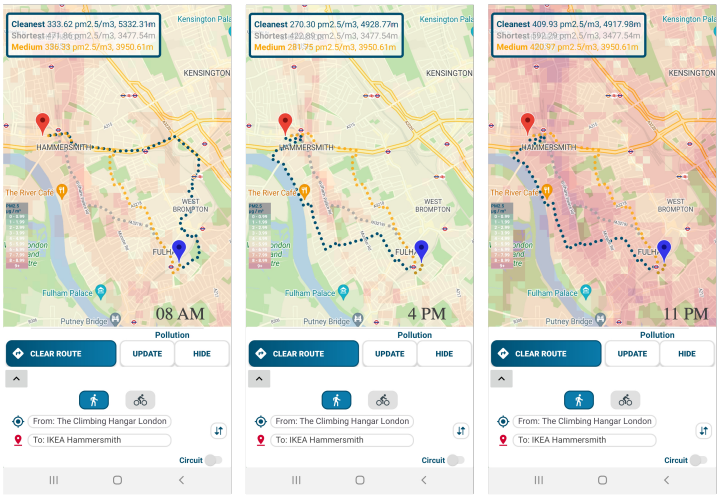

Figure 6.10: Evolution of a walking route in a single day in a heavier polluted area [Route B] at 8 AM, 4 PM, and 11 PM.

the distance in the cleanest and medium routes, in some instances, does not vary significantly. This can be explained by the resolution of the grid. Nonetheless, in Figure 6.10 it can be observed how the route traverses through less polluted areas such as the outskirts of the river, compared to the heavier polluted street paths.

Route	**Hour**	**Cleanest**$_p$	**Cleanest**$_d$	**Shortest**$_p$	**Shortest**$_d$	**Medium**$_p$	**Medium**$_d$
Route A	8 AM	280.22	4996	447.25	3949	299.85	4465
Route A	4 PM	238.44	4952	369.86	3949	250.71	4465
Route A	11 PM	365.19	4996	571.61	3949	391.08	4465
Route B	8 AM	336.62	5333	471.86	3477	363.33	3950
Route B	4 PM	270.30	4928	422.89	3477	281.75	3950
Route B	11 PM	409.97	4917	592.29	3477	420.97	3950

Table 6.3: Walking Route values for different trips at different hours of the day, where pollution is the total Sum of Pollutants measured in $PM_{2.5}$.

Lastly, the visualization for circuits follow a similar process for its generation as the point A to B route generation, and an example circuit, with the same start and ending points, as well as target distance D_t can be observed in Figure 6.11. In this context, it can be highlighted how the proposed circuits are enhanced by the pruning of closed streets.

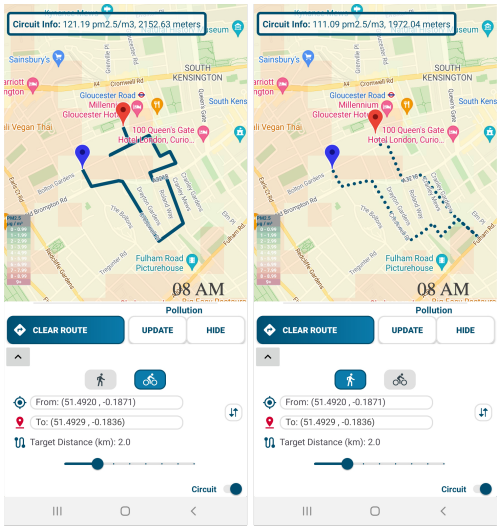

Figure 6.11: Example circuits created by cycling (right) and walking (left) and associated metrics shown in the App interface.

6.4.1 Mobile App Metrics

Finally, some relevant metrics of the application are disclosed. For the algorithm running time, a set of 15 route generation tests per mode was performed and averaged, and they refer to the complete process of running the algorithm and displaying the route on the screen. All metrics come from the log window. All testing was done on a mobile phone Galaxy A32 SM-A32.

Setting	Value
Apk size	10.1 MB
Loading Time	3.772 (s)
Get Route (walk)	249 (ms)
Get Route (bike)	751 (ms)
Get Route (circuit walk)	133 (ms)
Get Route (circuit bike)	106 (ms)

Table 6.4: Mobile App characterization metrics, where loading time refers upon initialization.

Albeit its initialization process, it is worth noting the fast response in route creation, where, in average, all routes generated take less than 1 second to be computed and displayed on screen. This efficiency can largely be attributed to the storage of the graphs as system files.

Chapter 7

Conclusions & Recommendations for Future Work

The core of this project rests upon the in-app execution of algorithms, which, although it provides swift responses, brings its very own challenges. These challenges primarily focus on computational constraints, which are a direct consequence of having to store graph information within the application itself, or as system files. This section develops on these challenges, proposes potential solutions, and suggests further exploration on the subject of cleanest route generation.

Firstly, the **Computational Constraints** the memory and processing requirements for managing large graph files within the system posed significant challenges. This restricted the scope of the project, as incorporating larger graphs often led to performance issues within the app such as lag and frame skipping. To improve these issues, there's the need to fasten the processing techniques or optimize graph loading. This could enhance the usability of the graphs as system files. Moreover, augmenting the grid size or refining the pollution resolution could help in providing cleaner and more precise route suggestions.

Secondly, for the **Graph Weight Assignments** the current system uses pollution values between two nodes to determine the edges' weights. To ensure a more accurate representation of pollution levels, there's an opportunity to reassess the weight assignment strategy. Considering pollution values across a broader range of adjacent nodes or evaluating multiple neighbouring nodes could lead to a more reliable weight calculation.

Thirdly, the **Algorithmic Refinement for Circuit Creation**, the current circuit generation process with BNN, could be further optimized. In this work, a tunning process

was followed to find a more suitable f value, instead of using the existing haversine and shortest route length ratio during the remaining distance budget computations, yet it still shows an overcalculation of the final distance for the circuits created. Further testing with this value would enable the algorithm to better approximate the desired target. Furthermore, exploring other arc-orienteering algorithms might also offer other valuable insights.

Fourthly, for the **Optimization of Weighted Dijkstra**, the weighted Dijkstra approach currently works as a trade-off mechanism in scenarios where significant length differences exist between the shortest and cleanest routes. Examining different heuristics could propose routes that not just balance but optimize the goals of route length and cleanliness.

Fifthly, for the **Comparative Analysis Limitations** the comparison between the real routes and the modelled ones indicated certain limitations, as detailed in the previous chapter. A more extensive dataset covering a broader range of routes could provide more insightful conclusions on the challenges of comparing pollution exposure.

Lastly, the **Data-Pipeline and Real-time Predictions** time-constrains prevented the development of a data-pipeline designed for real-time predictions. The absence of up-to-the-hour data from public sensor networks further exacerbated this challenge. Further effort to build and refine the data-pipeline is required, yet securing up-to-date data from public sensor networks remains a limitation.

In conclusion, while this project has made significant advancements in its objectives, the path forward offers opportunities for refinement and exploration. The recommendations provided in this section aim to guide future efforts towards more robust and insightful outcomes.

Bibliography

[1] D. K. Arvind, Janek Mann, Andrew Bates, and Konstantin Kotsev. The AirSpeck Family of Static and Mobile Wireless Air Quality Monitors. In *2016 Euromicro Conference on Digital System Design (DSD)*, pages 207–214. IEEE, 8 2016.

[2] Richard Atkinson, Sujin Kang, Ross Anderson, Inga Mills, and Heather Walton. Epidemiological time series studies of pm2.5 and daily mortality and hospital admissions: a systematic review and meta-analysis. *Thorax*, 69(7):660–665, 2014.

[3] Paper Authors and Tuuli Paper Author Roles. The Green Paths route planning software for exposure-optimised travel. Technical report.

[4] Geoff Boeing. OSMnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks. *Computers, Environment and Urban Systems*, 65:126–139, sep 2017.

[5] Andrés Bronfman, Vladimir Marianov, Germán Paredes-Belmar, and Armin Lüer-Villagra. The maximin hazmat routing problem. *European Journal of Operational Research*, 241(1):15–27, 2015.

[6] K.I. Cervantes-Sanmiguel, M.V. Chavez-Hernandez, and O.J. Ibarra-Rojas. Analyzing the trade-off between minimizing travel times and reducing monetary costs for users in the transit network design. *Transportation Research Part B: Methodological*, 173:142–161, 2023.

[7] Cross River Partnership and King's College London. Clean Air Walking Routes. https://crossriverpartnership.org/projects/clean-air-walking-routes/. [Online; accessed: 2023-August-18].

[8] Bhavya Deep. Determination of least polluted route using dijkstra's algorithm. *International Journal of Environmental Science and Technology*, pages 1–10, 01 2023.

[9] Daniel Delling, Peter Sanders, Dominik Schultes, and Dorothea Wagner. *Engineering Route Planning Algorithms*, pages 117–139. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[10] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[11] Daniel Duque, Leonardo Lozano, and Andrés L. Medaglia. An exact method for the biobjective shortest path problem for large-scale road networks. *European Journal of Operational Research*, 242(3):788–797, 2015.

[12] Enhao Sun. Least Polluted Routes: algorithms and application. Master's thesis, University Of Edinburgh, Edinburgh, 2019.

[13] Keivan Ghoseiri and Behnam Nadjari. An ant colony optimization algorithm for the bi-objective shortest path problem. *Applied Soft Computing*, 10(4):1237–1246, 2010. Optimisation Methods Applications in Decision-Making Processes.

[14] Aldy Gunawan, Hoong Chuin Lau, and Pieter Vansteenwegen. Orienteering problem: A survey of recent variants, solution approaches and applications. *European Journal of Operational Research*, 255(2):315–332, 2016.

[15] Ting Hua and Noraini Abdullah. Weighted sum-dijkstra's algorithm in best path identification based on multiple criteria. *Journal of Computer Science Computational Mathematics*, pages 107–113, 12 2018.

[16] Imperial College London. Health assessment across biological length scales for personal pollution exposure and its mitigation (INHALE). https://gtr.ukri.org/projects?ref=EP2FT0031892F1/tabOverview, 2019. [Online; accessed: 2023-August-18].

[17] Jamie Welsh. Spatial Estimation of PM2.5 Using Transfer Learning. Master's thesis, University Of Edinburgh, Edinburgh, 2023.

[18] Philip J Landrigan, Richard Fuller, Nereus J R Acosta, Olusoji Adeyi, Robert Arnold, Niladri (Nil) Basu, Abdoulaye Bibi Baldé, Roberto Bertollini, Stephan Bose-O'Reilly, Jo Ivey Boufford, Patrick N Breysse, Thomas Chiles, Chulabhorn Mahidol, Awa M Coll-Seck, Maureen L Cropper, Julius Fobil, Valentin Fuster, Michael Greenstone, Andy Haines, David Hanrahan, David Hunter, Mukesh

Khare, Alan Krupnick, Bruce Lanphear, Bindu Lohani, Keith Martin, Karen V Mathiasen, Maureen A McTeer, Christopher J L Murray, Johanita D Ndahimananjara, Frederica Perera, Janez Potočnik, Alexander S Preker, Jairam Ramesh, Johan Rockström, Carlos Salinas, Leona D Samson, Karti Sandilya, Peter D Sly, Kirk R Smith, Achim Steiner, Richard B Stewart, William A Suk, Onno C P van Schayck, Gautam N Yadama, Kandeh Yumkella, and Ma Zhong. The lancet commission on pollution and health. *The Lancet*, 391(10119):462–512, 2018.

[19] LondonAir. About LondonAir. https://www.londonair.org.uk/LondonAir/Default.aspx, 2023. [Online; accessed: 2023-April-23].

[20] Ying Lu and Cyrus Shahabi. An arc orienteering algorithm to find the most scenic path on a large-scale road network. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '15, New York, NY, USA, 2015. Association for Computing Machinery.

[21] Hagar Mahmoud and Nadine Akkari. Shortest path calculation: A comparative study for location-based recommender system. In *2016 World Symposium on Computer Applications Research (WSCAR)*, pages 1–5, 2016.

[22] Dimitrios Michail, Joris Kinable, Barak Naveh, and John V Sichi. Jgrapht – a java library for graph data structures and algorithms, 2020.

[23] John E. Mitchell. *Integer programming: branch and cut algorithmsInteger Programming: Branch and Cut Algorithms*, pages 1643–1650. Springer US, Boston, MA, 2009.

[24] OpenStreetMap Foundation. OpenStreetMap. https://www.openstreetmap.org/about. [Online; accessed: 2023-August-18].

[25] Pallets Team. *Flask Documentation*, 2021.

[26] E. Roghanian and Z. Shakeri Kebria. The combination of topsis method and dijkstra's algorithm in multi-attribute routing. *Scientia Iranica*, 24(5):2540–2549, 2017.

[27] Madeleine Santiago. Informatics project proposal, 2023. School of Informatics, University of Edinburgh.

[28] Sitthinut Kumpalanuwat. The Healthy City-tour Route Planner for cyclists in the area of central Edinburgh. Master's thesis, University Of Edinburgh, Edinburgh, 2019.

[29] Stefan Ivanov. Sensing Spaces: Least Polluted Route. Master's thesis, University Of Edinburgh, Edinburgh, 2018.

[30] Jason G. Su, Meghan Winters, Melissa Nunes, and Michael Brauer. Designing a route planner to facilitate and promote cycling in metro vancouver, canada. *Transportation Research Part A: Policy and Practice*, 44(7):495–505, 2010.

[31] UK Air. Automatic Urban and Rural Network (AURN). https://uk-air.defra.gov.uk/networks/network-info?view=aurn, 2023. [Online; accessed: 2023-April-23].

[32] Pieter Vansteenwegen, Wouter Souffriau, and Dirk Van Oudheusden. The orienteering problem: A survey. *European Journal of Operational Research*, 209(1):1–10, February 2011.

[33] C. Verbeeck, P. Vansteenwegen, and E.-H. Aghezzaf. An extension of the arc orienteering problem and its application to cycle trip planning. *Transportation Research Part E: Logistics and Transportation Review*, 68:64–78, 2014.

[34] World Health Organization. WHO global air quality guidelines: particulate matter (PM2.5 and PM10), ozone, nitrogen dioxide, sulfur dioxide and carbon monoxide. *World Health Organization*, pages xxi–273, 2021.

[35] Yu-Fei Xing, Yue-Hua Xu, Min-Hua Shi, and Yi-Xin Lian. The impact of pm2.5 on the human respiratory system. *Journal of Thoracic Disease*, 8(1), 2016.

[36] Murat Yener and Onur Dundar. *Android Application Development With Android Studio*, pages 45–79. 04 2017.

[37] Zoë Petard. High Resolution Spatial Predictions of Air Pollution Levels in Edinburgh. Master's thesis, University Of Edinburgh, Edinburgh, 2018.

[38] Bin Zou, Shenxin Li, Zhong Zheng, Benjamin F. Zhan, Zhonglin Yang, and Neng Wan. Healthier routes planning: A new method and online implementation for minimizing air pollution exposure risk. *Computers, Environment and Urban Systems*, 80:101456, 3 2020.