Smart-Routing Web App: A Road Traffic Eco-routing Tool Proposal for Smart Cities

José R. Lozano-Pinilla[1][0000−0003−0759−1635], Ivan Sánchez Cordero[0009−0008−1096−8329], and Cristina Vicente-Chicote[1][0000−0002−9553−3461]

Quercus SEG, University of Extremadura, Cáceres, Spain {joserralp, cristinav}@unex.es

Abstract. The use of routing services has witnessed a notable surge in recent years. While most of them provide users with the shortest and the fastest routes, only a few of them provide information about the most eco-friendly route or gather information about the vehicle or the user preferences. Eco-routing has demonstrated its potential to significantly reduce both fuel consumption and Greenhouse Gas Emissions (GGE). However, most of the routing applications supporting this feature do not consider the specific car features, the road slope or the traffic conditions, providing only a rough estimation of the fuel consumption (mainly based on travel distance and type of fuel). Integrating such additional information would result in more flexible and powerful routing applications, allowing end-users to prioritize different features (travel time, distance, fuel consumption, etc.) according to their needs or preferences. In this context, we propose an easy-to-configure smart-routing web framework, providing end-users with alternative routes for their trips, including the most common ones (minimum distance and minimum expected travel time) together with an eco-friendly route, computed in a more precise way than current routing services.

Keywords: Eco-Routing · Route Optimization · Smart Cities.

1 Introduction

A few decades ago, the only way to plan travel routes was by using physical road maps. Currently, navigation apps (installed in our mobile phones or even integrated in our cars) help us plan and carry out our trips in a more user-friendly and sustainable way, as maps are automatically updated (there is no need to regularly buy new versions of printed maps) and include plenty of additional useful information.

Currently, navigation apps increasingly integrate features related with smart mobility, including eco-route planning, real-time traffic status, or information about accidents or road works, among others. These features not only help individual drivers to plan and carry out their trips, but also aim at contributing to global traffic management, one of the key aspects in *Smart Cities*. Some of these smart features (e.g., traffic status) build on information automatically gathered

from users' mobile devices. This is the case of Google Maps [1]: one of the leading navigation apps today, and a precursor to many others. Google Maps provides users with many valuable features: it is free, easy to configure and use, and is available both as a web and a mobile application. However, some users report limited offline functionality, excessive battery consumption, and many privacy issues related with personal data collection, storage and exploitation. In this sense, the use of Internet of Things (IoT) technologies could help implementing more privacy-respectful navigation systems, as traffic information could be collected in an aggregated and anonymous way from different sensors placed in the cities' infrastructure (rather than from the users' devices). The use of this information for intelligent decision-making related to traffic management would contribute to achieve real Smart Cities. However, the integration of IoT devices in cities is not being as fast or uniform as promised, resulting in unreliable performance and lack of information for many places.

One of the most recent features included in Google Maps is eco-routing. This concept, coined by Ericsson et al. [2] in 2006, aims at computing the route that minimizes the energy required to complete a trip, taking into account several factors such as the current status of the traffic, the road topology (e.g., slope, road type, diversions, etc.) and the specific features of the vehicle (e.g., weight, fuel type, etc.), among others. Reducing vehicles' energy consumption implies not only reducing the cost of the trip, but also the Greenhouse Gas Emissions (GGE). Besides, specifically to Electric Vehicle (EV) drivers, eco-routing also reduces "range anxiety", i.e., drivers' anxiety for not being able to complete their trip before the vehicle runs out of battery. Despite these advantages, eco-routes tend to be longer in terms of travel time since, e.g., they may consider roads with less slope that may require traveling longer distances. In fact, the study carried out in Japan by Kono et al. [3] concluded that eco-routes, compared to shortest routes, reduce fuel consumption in about 9%, while they increase travel time also in 9%. Despite the longer travel times, Zeng et al. highlight in [4,5] the benefits of choosing eco-routes compared to shortest or fastest routes.

Eco-route calculation is a computation-intensive process involving big amounts of input data. Although Google Maps supports this feature, it offers just a rough estimation of eco-routes, namely based on fuel type and travel distance. In this context, the Smart-Routing Web App presented in this paper aims at computing eco-routes in a more accurate way, while keeping the process efficient and preserving users' privacy. Apart from eco-routes, the application will also provide drivers with the fastest and shortest routes, allowing them to choose among the three of them according to their needs or preferences.

The rest of the paper is organized as follows. Firstly, section 2 reviews related research focused on eco-routing. Secondly, section 3, outlines the architecture of the proposed web application, describing its main components. Then, section 4 describes how road topology information is obtained, processed and used to compute the eco-routes according to the workflow then detailed in section 5. Finally, Section 6 draws some conclusions and outlines future works.

2 Related works

Eco-routing is a hot research topic today due to its relevance in smart mobility: one of the pillars of Smart Cities. Smart mobility aims at reducing travel times and congestion, improving network and traffic management, increase safety by preventing accidents, and ensure environmental and economic sustainability.

Many research works focused on eco-routing can be found in the literature. Zeng et al. [6] develop an eco-routing algorithm for navigation systems that finds the route with the minimum fuel consumption satisfying on-time arrival requirements. Aguiar et al. [7] define MobiWise, an eco-routing decision support system that leverages Internet of Things (IoT) technologies, real traffic data and a calibrated digital twin, being limited to a middle-sized European city as Porto (Portugal). Some works focus on reducing emissions and air pollution. For instance, Vamshi et al. [8] propose a dynamic route planning algorithm that distributes traffic density in real-time to low-density traffic areas, trying to minimize the number of congested junctions by uniformly distributing traffic in order to reduce air pollution, available only with traffic information on the Surat area in Gujarat (India). Similarly, Ghaffari et al. [9] introduce a novel algorithm to find the shortest route based on traffic congestion and air quality, not considering road topology or vehicle features.

Internet of Vehicles (IoV), an evolution of the Vehicular Ad-hoc Networks (VANETs), is another active research field in which it is possible to find some works focused on eco-routing. For instance, Xiaofeng [10] introduces a hybrid genetic algorithm and social spider optimization algorithm for an energy-aware routing schema to optimize traffic congestion. Also in this line, Alfaseeh et al. [11] propose a multi-objective eco-routing system for Connected and Autonomous Vehicles (CAVs). Both IoV and CAVs assume that vehicles are equipped with sensors, software, and other technologies to collect and exchange information over the Internet with other vehicles or smart devices. However, these technologies are not expected to be available in most private vehicles until 2040 [12,13].

Finally, among the navigation systems currently supporting eco-route calculation it is worth highlighting Google Maps [14] and EmiLa App [15]. The former was first introduced by Google in the United States in October 2021. Nowadays, it is available in many other countries, more than 40 in Europe [16], including Spain, France, Germany and the UK. Google Maps allows drivers to configure the type of engine of their vehicles: diesel, gasoline, electric or hybrid and, according to that, it displays the eco-route and a rough estimation of the fuel savings. However, as previously mentioned, the eco-route proposed by Google Maps seems not to consider, e.g., relevant changes in the route slope, which may drastically affect energy consumption. On the other hand, EmiLa App integrates life-cycle assessments of various means of transportation into classic route planning algorithms by considering different sustainability factors in addition to typical metrics such as distance, travel times or cost. This application motivates users to choose eco-routes through gamification. However, the criteria for eco-route selection and emission calculation are not clearly defined and it seems to be still a theoretical artifact rather than an actual product.

Despite the advances achieved in the eco-routing field, most of the reviewed works have important limitations either in the geographical areas of application, in the technologies required for their implementation or in the incomplete information being used for the calculations. In response to these limitations, our proposal aims to address worldwide eco-route calculation using currently available technologies and accurate topology information.

3 Proposed architecture

The software architecture of the Smart-Routing Web Application proposed in this paper gathers several components. These components are deployed and connected in a loosely coupled way, making it easier to replace any of them with other services offering similar functionalities. The architecture of the proposed framework is shown in Fig. 1, along with its components and their relationships. Each of these components is described below, along with their core functionality and the technologies involved.

– **Routing Services (RS)**. This component currently gathers three third-party routing services, although it could be easy extended to support new (either third-party or self-made) ones. The three services currently supported build on OpenStreetMap, as a baseline for route calculation: (1) OpenRouteService [17] (ORS), which provides global spatial services such as route calculation or time-distance matrices calculated using only public open-source data; (2) Open Source Routing Machine [18,19] (OSRM), a high-performance routing engine for calculating shortest and fastest routes in road networks, among other functionalities; and (3) GraphHopper [20,21], an open-source

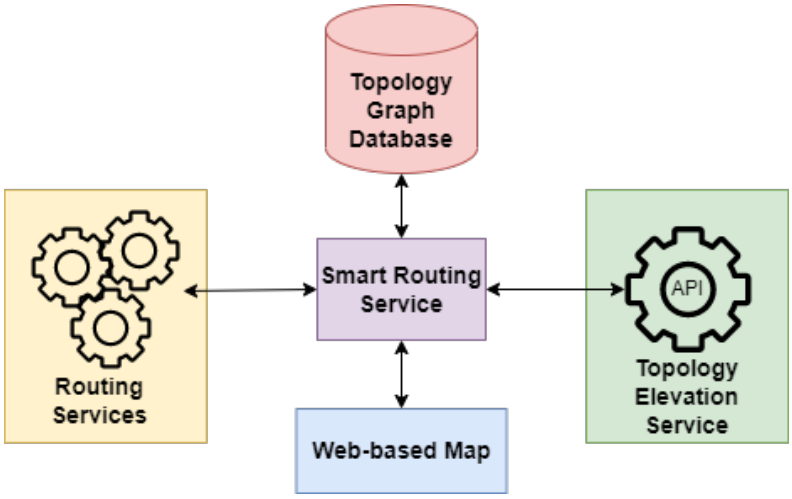

Fig. 1: Smart Routing Web Application architecture.

fast and memory-efficient routing engine, which allows users to calculate the distance, time, turn-by-turn instructions, and road features for any route between two or more points. All these services calculate routes based on different metrics, allowing users to easily retrieve them using different programming languages. Although all these services are third-party, some of them have been deployed locally using Docker containers.

– **Topology Elevation Service (TES)**. Since one of the main contributions of this framework is to accurately calculate eco-routes, knowledge of the road network topology is essential due to its direct impact on fuel consumption. There are several services that allow us to determine the elevation (altitude) of any GPS coordinate on Earth such as (1) Open-Elevation API [22], a free and open-source API that allows retrieving elevation information for geographical coordinates; (2) Google Elevation API [23], a pay-per-request API that returns elevation data for a given location or path; and (3) OpenTopoData [24], a free REST API for elevation data that can be deployed both locally or remotely. We chose the later as it allows users to easily select different data sources and load them in the API. The process for loading and processing geospatial and topology data is later described in Section 4.
– **Web-Based Map (WBM)**. This component serves as an interface between the SRS component and the end-users, allowing them to select the source and destination of their trip and view the different routes calculated between them. The application is based on an interactive map, meaning that users can select locations directly clicking on it and interact with the output routes to retrieve additional information, such as the distance or the estimated travel time or fuel consumption. Additionally, this application provides turn-by-turn directions for the selected routes, offering specific road information at each route segment. Figure 2 shows a mock-up of the web application with all its detailed features. This component has been developed using the Flask micro-framework, along with Python and Jinja to handle user interactions, and the interactive and easy to customize maps provided by OpenStreetMap.

Fig. 2: Smart Routing Web App mock-up

- **Topology Graph Database (TGD)**. In order to store the different routes and efficiently perform operations on them, we decided to use a graph-oriented database. Among the different options considered (e.g., ArangoDB or RedisGraph), finally Neo4J was chosen because it supports both directed and undirected graphs, and offers several libraries and predefined route optimization algorithms such as A*, shortest path, and Yen's k-shortest paths, among others.
- **Smart Routing Service (SRS)**. This component is the core of the framework as it interconnects all other services. The SRS is responsible for retrieving the different routes provided by the RS, processing and storing them in the TGD and plotting them graphically in the WBM. Route processing involves several steps: (1) retrieving the full routes; (2) splitting them into different route segments and nodes; and (3) retrieving and calculating the topology information for these segments, including their slope, number of lanes or maximum speed, among other features. For the last step, the Overpass API [25], an OpenStreetMap read-only service that allows users to retrieve OSM-map-related information, is used. The SRS component has been developed using Python and Flask, along with several additional libraries and packages.

Some of the external services used in the framework can also be deployed locally, which facilitates the development and testing processes. These local services are deployed using Docker containers and the Docker-Compose orchestration tool, as shown in Fig. 3, where the architecture of the local components, their properties and relations are described using the *Containers Modeling Tool (CML)* [26]: a Docker-Compose modeling tool developed by the authors. Components used remotely are not represented in the model, while all local containers have a related volume, where their functionality information is stored.

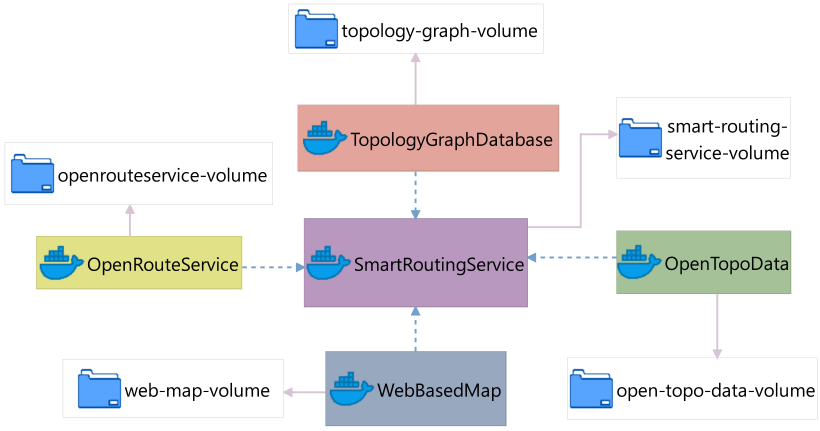

Fig. 3: Docker-Compose schema of the Smart Routing Web App.

Despite its benefits, local deployment has also some limitations. For instance, some of the routing services do not provide all the features supported by their online versions, e.g., the route step descriptions or the topology database may be limited to the information stored on the local device, even when using a worldwide dataset. These limitations have been considered and addressed in the framework by obtaining any missing information from other external sources.

4 Topology information

The topology information obtained by the TES component is very valuable as it allows calculating optimal eco-routes in a more precise way. In the proposed Smart-Routing Web Application, one the sources providing topology information is the OpenTopoData API, which is deployed locally and allows loading almost any kind of geospatial dataset (ASTER, ETOPO1, SRTM both 30m and 90m, Mapzen, etc.). The framework loads and uses several datasets, including the one provided by the NASA's Shuttle Radar Topography Mission (SRTM) [27]. This dataset covers almost all of the earth's surface (from -60 to 60 degrees latitude) with a precision of 1 arc-second (30 meters at the equator). As this dataset covers a huge extension, it is divided into several chunks of information, each one covering different areas, as shown in Fig. 4, which displays two adjacent SRTM30m files.

The full dataset is publicly available worldwide for testing purposes, consisting of 14,297 files with an estimated total size of 345 GB. As the full dataset could not be stored and managed efficiently on a local device, we restricted the data downloaded (using the 30-meter SRTM Tile Downloader tool [28]) to Spain (88 files, totaling around 2 GB). This information is loaded into the OpenTopoData API service and used to test the framework, however, it is easy to modify.

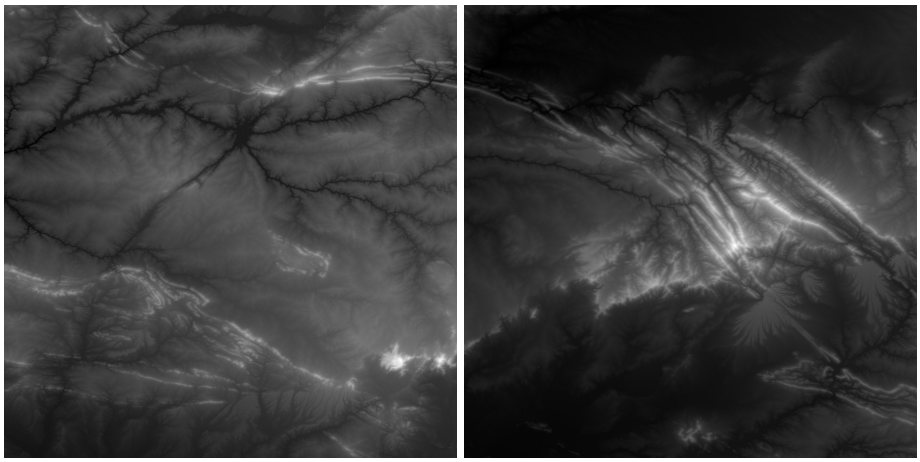

Fig. 4: SRTM30m file examples

Several tests were performed to determine the accuracy of the information provided by this dataset, resulting in a very low error of around a few meters. These tests involved comparing the altitude obtained from different datasets and APIs, as well as conducting empirical experiments in which the authors physically visited several locations to verify the accuracy of the data.

The geospatial information retrieved by the system was subsequently used to calculate metrics related to the different routes and their segments, such as distances and slopes. In order to calculate the slope of a given segment, information about the distance between its two coordinates and their respective altitudes is required (see Equation 1). The resulting slope can be either positive (ascending) or negative (descending) and is measured in percentage (%).

$$slope = \frac{destination\_altitude - source\_altitude}{distance} * 100 \tag{1}$$

Regarding road topology, different metrics were considered such as road speed limit, number of lanes or road types (e.g., highways, streets, secondary roads), among others. This information was retrieved from the Overpass API provided by OpenStreetMap, using the Overpass Query Language (Overpass QL).

Figure 5 shows an example in which different routes between a source and a destination have been calculated. All routes are represented in the Topology Graph Database (TGD) as a directed graph. Each colored bubble represents an OSM node and each link between two nodes stores the properties of the road connecting them. In this example four routes were obtained. The shortest one is displayed in yellow, the fastest in blue and the eco-route in green. The fourth one, displayed in black, is a valid but not optimal route according to the selected criteria, i.e., shortest distance, lower travel time and lower fuel consumption. Table 1 displays these routes along with their paths and estimated metrics.

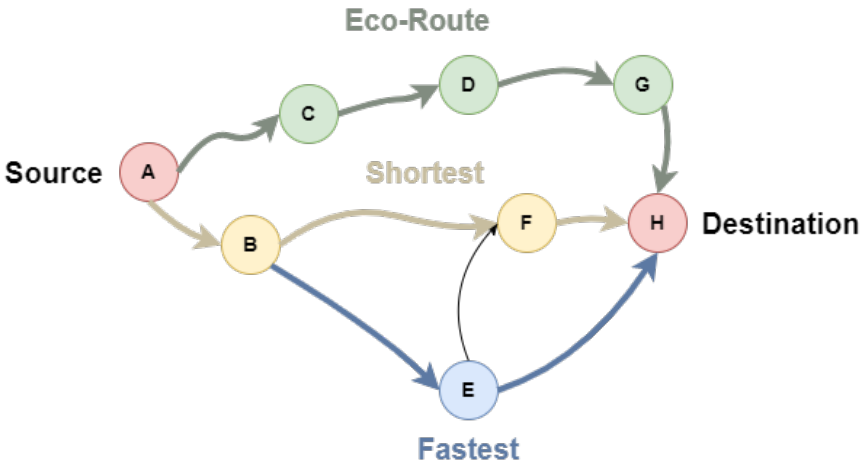

Fig. 5: Graph generated for the four routes retrieved by the SRS.

Table 1: Calculated routes. Path and related metrics: Distance (D), Estimated Travel Time (ETT), Estimated Fuel Consumption (EFC).

Route Id.	Path	D (meters)	ETT (minutes)	EFC (liters)
1	A, B, F, H	**10525** (*shortest*)	21,05	0,75
2	A, C, D, G, H	11212	26,91	**0.58** (*eco-route*)
3	A, B, E, H	10825	**18.56** (*fastest*)	0.69
4	A, B, E, F, H	10913	21.83	0.71

5 Workflow

The workflow of the proposed framework is illustrated in Fig. 6 and involves all its components. The process begins when the user selects the source and the destination of the trip using the WBM. Then, the SRS parses this input and sends the corresponding requests to the routing services supported by the RS component. Once they return the calculated routes, they are individually processed according to the following steps:

1. All the OSM nodes of each route are retrieved.
2. The road segments connecting these nodes are obtained.
3. The geospatial information for each node is retrieved using the TES.
4. For each road segment, its length, speed limit, number of lanes and restrictions are obtained.
5. The slope of each road segment is calculated using its length and the elevation of the two nodes it connects.
6. All the route information is stored in the TGD.

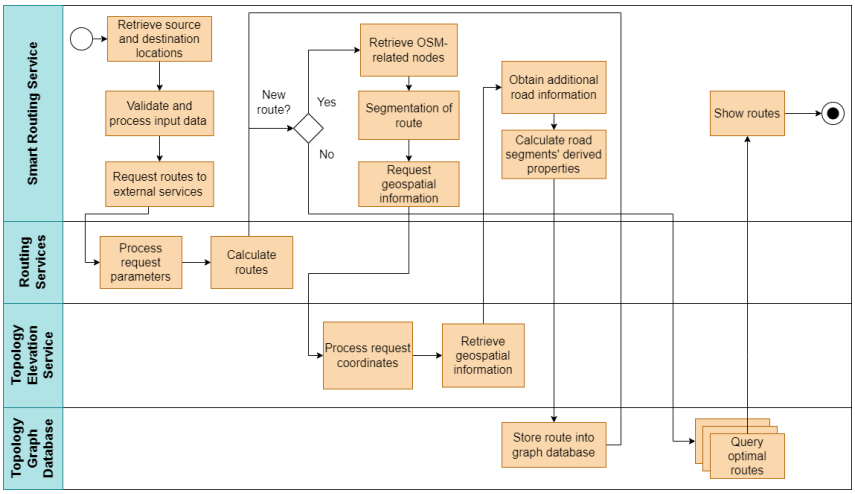

Fig. 6: Smart Routing Web App workflow.

Once all the routes have been successfully processed and stored in the database, three different queries allow calculating which one is shorter (minimum total distance), faster (minimum total travel time considering each segment length and speed limit), and more eco-friendly (less fuel consumption considering each segment length, speed limit and slope). Finally, the result of these queries is graphically displayed on the web-based map, allowing users to select the route that better fits their preferences or needs. Additionally, directions are provided for the selected route.

6 Conclusions and future works

The work presented in this paper introduces a Smart-Routing Web App aimed at helping users plan their road trips considering not only the fastest or the shortest routes (already provided by most navigation apps), but also the most eco-friendly one, i.e., the less fuel consuming (and thus, less contaminating) route. Although some of the existing route planning apps already provide users with eco-routes, their estimations about fuel consumption/savings do not account for important aspects such as the road slope. In order to cope with this limitation, the proposed framework builds on different route planning and topological data services to obtain more precise metrics on the different routes available for a given trip, namely: estimated distance, travel time and fuel consumption. In the paper, both the structure (loosely-coupled component-based architecture) and the behavior (workflow) of the developed framework have been presented, along with an illustrating example.

Building on this preliminary work we plan some future extensions, including: (1) increasing the precision of the travel time and the fuel consumption estimations by considering both the traffic conditions (e.g., using the information provided by the MapBox API [29]) and the specific features of the vehicle used for the trip (e.g., using FASTSim: the Future Automotive Systems Technology Simulator [30]). Traffic conditions may significantly affect the speed profile in some segments and, as a consequence the travel time and the fuel consumption, e.g., under congestion. Similarly, the specific features of the vehicle (weight, idle consumption, etc.) have a significant impact on fuel consumption and cannot be neglected; (2) considering city-specific limitations such as zero-emission zones; (3) replacing the graph database instance by an in-memory graph projection to enable route caching and shorten query execution times; and (4) integrating the proposed tool into an API for easy remote access.

Acknowledgements. This work was supported by Project TED2021-132696B-I00, funded by MCIN/ AEI /10.13039/501100011033/ and by ERDF A way to build Europe. José R. Lozano-Pinilla thanks the Junta de Extremadura for its Recovery, Transformation and Resilience Plan (funded by Next Generation EU), currently supporting him with an INVESTIGO contract.

References

1. Google Maps, Google, https://www.google.es/maps/. Last accessed 14 July 2023.
2. Ericsson, E., Larsson, H., Brundell-Freij, K.: Optimizing route choice for lowest fuel consumption – Potential effects of a new driver support tool, Transportation Research Part C: Emerging Technologies, vol. 14, pp. 369–383. (2006). https://doi.org/10.1016/j.trc.2006.10.001.
3. Kono, T., Fushiki, T., Asada, K., Nakano, K.: Fuel consumption analysis and prediction model for "Eco" route search. In: 15th World Congress on Intelligent Transport Systems and ITS America's 2008 Annual Meeting. (2008) https://trid.trb.org/view/902235
4. Zeng, W., Miwa, T., Morikawa, T.: Prediction of vehicle CO2 emission and its application to eco-routing navigation, Transportation Research Part C: Emerging Technologies, vol. 68, pp. 194–214. (2016). https://doi.org/10.1016/j.trc.2016.04.007.
5. Zeng, W., Miwa, T., Morikawa, T.: Application of the support vector machine and heuristic k-shortest path algorithm to determine the most eco-friendly path with a travel time constraint, Transportation Research Part D: Transport and Environment, vol. 57, pp. 458–473. (2017). https://doi.org/10.1016/j.trd.2017.10.001.
6. Zeng, W., Miwa, T., Morikawa, T.: Eco-routing problem considering fuel consumption and probabilistic travel time budget, Transportation Research Part D: Transport and Environment, vol. 78. (2020) https://doi.org/10.1016/j.trd.2019.102219.
7. Aguiar, A., Fernandes, P., Guerreiro, A. P., Tomás, R., Agnelo, J., Santos, J. L., Araújo, F., Coelho, M. C., Fonseca, C. M., d'Orey, P. M., Luís, M., Sargento, S.: MobiWise: Eco-routing decision support leveraging the Internet of Things, Sustainable Cities and Society, vol. 87. (2022) https://doi.org/https://doi.org/10.1016/j.scs.2022.104180.
8. Vamshi, B. and Prasad, R. V.: Dynamic route planning framework for minimal air pollution exposure in urban road transportation systems, 2018 IEEE 4th World Forum on Internet of Things (WF-IoT), Singapore, pp. 540–545. (2018) https://doi.org/10.1109/WF-IoT.2018.8355209.
9. Ghaffari, E., Rahmani, A. M., Saberikamarposhti, M. and Sahafi, A.: An Optimal Path-Finding Algorithm in Smart Cities by Considering Traffic Congestion and Air Pollution, in IEEE Access, vol. 10, pp. 55126–55135. (2022) https://doi.org/10.1109/ACCESS.2022.3174598.
10. Xiaofeng, S.: Improved Energy-efficient Routing Architecture for Traffic Management System Using a Hybrid Meta-heuristic Algorithm in Internet of Vehicles. (2022). https://doi.org/https://doi.org/10.3233/JHS-222003
11. Alfaseeh, L., Djavadian, S., Tu, R., Farooq, B. and Hatzopoulou, M.: Multi-objective Eco-routing in a Distributed Routing Framework, 2019 IEEE International Smart Cities Conference (ISC2), Morocco, pp. 747–752. (2019) https://doi.org/10.1109/ISC246665.2019.9071744.
12. Rydzewski, A., and Czarnul, P.: "Recent advances in traffic optimisation: systematic literature review of modern models, methods and algorithms", IET Intelligent Transport Systems, vol. 14, pp. 1740-–1758. (2020) https://doi.org/10.1049/iet-its.2020.0328.
13. Winkle, T.: "Safety Benefits of Automated Vehicles: Extended Findings from Accident Research for Development, Validation and Testing", Autonomous Driving, Berlin. (2016). https://doi.org/10.1007/978-3-662-48847-8_17
14. Use eco-friendly routing on your Google Maps app, Google, https://support.google.com/maps/answer/11470237?hl=en. Last accessed 1 April 2023.

15. Heckmann, R., Gaspers, L., Schönberger, J.: Development of an Eco-Routing App to Support Sustainable Mobility Behaviour. Innovations for Metropolitan Areas. (2022). https://doi.org/10.1007/978-3-662-60806-7_20
16. Google Maps is expanding its eco-friendly navigation feature to Europe, TechCrunch, https://techcrunch.com/2022/09/06/google-maps-is-expanding-its-eco-friendly-navigation-feature-to-40-more-countries/. Last accessed 1 July 2023.
17. OpenRouteService, https://openrouteservice.org/. Last accessed 10 July 2023.
18. Open Source Routing Machine, OSRM Project. https://project-osrm.org/. Last accessed 20 July 2023.
19. Open Source Routing Machine, GitHub. https://github.com/Project-OSRM/osrm-backend. Last accessed 14 July 2023.
20. GraphHopper, GraphHopper. https://www.graphhopper.com/. Last accessed 23 July 2023.
21. GraphHopper, GitHub, https://github.com/graphhopper/graphhopper. Last accessed 26 July 2023.
22. Open-Elevation API, https://open-elevation.com/. Last accessed 10 July 2023.
23. Elevation API Google, https://developers.google.com/maps/documentation/elevation/overview. Last accessed 13 July 2023.
24. Open Topo Data, https://www.opentopodata.org/. Last accessed 30 June 2023.
25. Overpass API User's Manual, https://dev.overpass-api.de/overpass-doc/en/. Last accessed 20 June 2023.
26. Containers Modeling Language (CML), GitHub. https://github.com/elpiter15/CML. Last accessed 4 July 2023.
27. Shuttle Radar Topography Mission, NASA. https://www2.jpl.nasa.gov/srtm/. Last accessed 10 July 2023.
28. 30-Meter SRTM Tile Downloader, https://dwtkns.com/srtm30m/. Last accessed 4 June 2023
29. Mapbox API, https://docs.mapbox.com/api/overview/. Last accessed 20 July 2023.
30. FASTSim: Future Automotive Systems Technology Simulator, National Renewable Energy Laboratory (NREL). https://www.nrel.gov/transportation/fastsim.html. Last accessed 20 July 2023.