International Journal of Creative and Open Research in Engineering and Management
ISSN: **3108-1754 (Online)**
Volume 02 Issue 07 July-2026 | Impact Factor: 3.5

GreenHalo Maps: An AI-Powered Eco-Navigation System with Multi-Model Emission Prediction and Ant Colony Optimization for Sustainable Urban Routing

B. Uday Kiran
Student
Dept. of CSE (Data Science)
Geethanjali College of Engineering & Technology
22r11a6791@gcet.edu.in

A. Prudhvi Raj
Student
Dept. of CSE (Data Science)
Geethanjali College of Engineering & Technology
23r15a6709@gcet.edu.in

S. Rushi Govind
Student
Dept. of CSE (Data Science)
Geethanjali College of Engineering & Technology
22r11a67c1@gcet.edu.in

Dr. M. Srinivas
Associate Professor
Dept. of CSE (Data Science)
Geethanjali College of Engineering & Technology
Madhirasrinivas.cse@gcet.edu.in

Abstract

Sustainable urban mobility has become a critical challenge as vehicular emissions continue to be a primary driver of air pollution and climate change. This paper presents GreenHalo Maps, a lightweight, real-time eco-navigation system that integrates machine learning-based carbon emission prediction with Ant Colony Optimization (ACO) to recommend the most environmentally responsible driving route among available alternatives. Unlike conventional navigation systems that optimize solely for speed or distance, GreenHalo Maps evaluates each candidate route across five environmental parameters: trip distance, elevation gain, traffic congestion index, fuel type, and vehicle category. Two regression models — a Random Forest Regressor (250 estimators) and a LightGBM Regressor (500 estimators) — predict CO_2 emission in kilograms per route with R^2 scores of 0.96 and 0.95 respectively, trained on a continuously growing dataset of 1,062 real route records collected via the Google Maps Directions and Elevation APIs. An ACO algorithm operating on a coordinate graph built from decoded route polylines then identifies the minimum-emission path by optimising pheromone trails over 40 iterations with 20 ants. Air quality at each route's midpoint is retrieved from the Google Air Quality API and surfaced as an AQI badge alongside emission figures. The system auto-retrains both models every 500 new route records logged to the training CSV, ensuring continuous model improvement without manual intervention. Deployed as a Flask web application with a Google Maps JavaScript frontend, GreenHalo Maps renders colour-coded polylines for up to three alternative routes, displays an interactive emissions comparison table, and highlights the ACO-optimised eco-route. The full pipeline installs from seven pip packages with no version conflicts across Python 3.8–3.12 and operates at interactive latency on commodity CPU hardware.

Key Terms—Eco-navigation, carbon emission estimation, ant colony optimization, random forest, LightGBM, Google Maps API, air quality index, sustainable routing, Flask, smart city.

I. Introduction

The transport sector accounts for approximately 24% of global CO_2 emissions from energy combustion, with road vehicles responsible for the largest share [1]. In rapidly urbanising regions such as India, where traffic congestion is endemic and mixed fuel-type fleets are common, route-level emission variation between parallel paths to the same destination can be substantial. Yet mainstream navigation systems — Google Maps, Apple Maps, Waze — continue to optimise primarily for travel time or distance, with only rudimentary eco-routing support and no per-route emission quantification visible to the user.

Intelligent eco-navigation systems have been proposed to address this gap, but prior systems face two recurring limitations. First, emission models are typically static lookup tables based on average vehicle class figures, ignoring dynamic factors such as real-time traffic congestion, route-specific elevation profile, and fuel type. Second, route optimisation is computationally expensive when using exact methods on city-scale graphs, leading researchers to either restrict to a pre-enumerated small set of candidate routes or to deploy cloud-side computation that introduces latency and data-privacy concerns [2].

In this paper, we present GreenHalo Maps, an end-to-end eco-navigation system that addresses both limitations. The emission estimation layer uses two independently trained machine learning regressors — Random Forest and LightGBM — fed with five dynamic features per candidate route. The route selection layer applies Ant Colony Optimization (ACO) on a coordinate graph derived from the polyline encodings of candidate routes provided by the Google Maps Directions API, efficiently exploring the solution space without exhaustive enumeration. Both layers are served by a lightweight Flask backend with a Google Maps JavaScript frontend, deployable on a standard laptop without GPU or cloud dependency.

The principal contributions of this paper are as follows:

1. A dual-model ML emission pipeline (Random Forest + LightGBM) trained on real API-sourced route data, achieving R^2 = 0.96 and 0.95 respectively across five predictive features.
2. An ACO-based route optimisation algorithm operating on a coordinate-level graph derived from Google Maps polylines, minimising estimated CO_2 emission cost.
3. A real-time air quality integration layer querying the Google Air Quality API at each route midpoint to surface AQI health categories alongside emission figures.
4. An automatic retraining pipeline that appends each user query to a training CSV and triggers model retraining every 500 records, enabling continuous improvement.
5. A fully functional Flask web application with a Google Maps JavaScript interface rendering colour-coded multi-route polylines and an emission comparison dashboard.

II. Related Work

A. Eco-Routing and Green Navigation

Eco-routing research has a two-decade history stemming from the fuel-optimal vehicle routing literature. Barth and Boriboonsomsin [3] demonstrated that eco-routing can reduce fuel consumption by 8–15% compared to fastest-path routing on urban networks. Frey et al. [4] showed that elevation and acceleration events contribute disproportionately to CO_2 output even on short-distance urban trips. Modern eco-routing implementations fall into trajectory-based approaches that require second-by-second speed profiles [5] and segment-based approaches assigning average emission rates to road segments [6]. GreenHalo Maps takes a route-level approach using route-aggregate features (total distance, total elevation gain, congestion index), trading fine granularity for deployment simplicity and compatibility with standard navigation APIs.

B. Machine Learning for Emission Estimation

Statistical and machine learning models have been applied to vehicle emission estimation since the MOVES [7] and COPERT [8] frameworks. Recent work applies gradient boosting regressors to emission prediction using GPS trace features, achieving MAE below 0.15 kg CO_2 per trip on taxi fleet datasets [9]. Random Forest models have been used for link-level emission prediction in urban scenarios [10], while LightGBM has demonstrated superior training speed on large-scale transport datasets [11]. The present work combines both architectures, presenting their independent predictions to the user for transparency rather than blending into a single opaque score.

C. Ant Colony Optimization for Routing

Ant Colony Optimization, introduced by Dorigo et al. [12], is a probabilistic metaheuristic inspired by ant foraging behaviour, widely applied to the Travelling Salesman Problem and vehicle routing. ACO has been adapted for road network shortest-path problems [13] and multi-objective vehicle routing with environmental objectives [14]. ACO's stochastic exploration makes it well-suited to discovering low-emission paths that deterministic greedy algorithms would miss. The implementation uses a lightweight single-objective ACO with 20 ants and 40 iterations, calibrated for interactive-latency execution on CPU hardware.

D. Air Quality in Navigation

Incorporating air quality into navigation decisions has received attention in personal health navigation literature [15]. Systems such as CleanRoute [16] route pedestrians and cyclists along lower-pollution corridors using sensor network AQI readings. GreenHalo Maps includes AQI primarily as an informational indicator for the driver, surfacing health context alongside emission figures rather than using it as a routing objective.

E. Comparison of ACO with Related Optimisation Techniques

Several route optimisation techniques have been applied to eco-routing and vehicle routing problems. Table IV provides a structured comparison of ACO against five widely studied alternatives across eight dimensions relevant to eco-navigation deployment: solution quality, convergence speed, support for multi-objective problems, computational complexity, adaptability to dynamic real-time inputs, ease of implementation, prior application to eco-routing, and suitability for deployment on commodity CPU hardware without GPU dependency.

Technique	Solution Quality	Convergence Speed	Multi-Objective	Complexity	Dynamic Inputs	CPU Deployable
ACO (proposed)	High	Moderate	Yes	$O(n^2 \cdot I \cdot A)$	Yes	Yes
Dijkstra / A*	Optimal	Fast	No	$O((V+E)\log V)$	Limited	Yes
Genetic Algorithm	High	Slow	Yes	$O(G \cdot P \cdot n)$	Limited	Yes
Simulated Annealing	High	Moderate	Limited	$O(n \cdot T)$	Limited	Yes
Particle Swarm Opt.	Moderate	Fast	Yes	$O(P \cdot n \cdot I)$	Limited	Yes
Reinforcement Lear	Very High	Very Slow	Yes	$O(S \cdot A \cdot E)$	Yes	GPU Pref.

Technique	Solution Quality	Convergence Speed	Multi-Objective	Complexity	Dynamic Inputs	CPU Deployable
ning						
Greedy / Heuristic	Low-Med	Very Fast	No	O(n logn)	Yes	Yes

TABLE I. COMPARISON OF ACO WITH RELATED OPTIMISATION TECHNIQUES (N=NODES, I=ITERATIONS, A=ANTS, G=GENERATIONS, P=POPULATION, T=TEMP. STEPS, S=STATES, E=EPISODES)

As shown in Table IV, Dijkstra and A* guarantee optimal shortest-path solutions but are inherently single-objective and provide no native mechanism for incorporating emission costs dynamically. Genetic Algorithms (GA) offer high solution quality for multi-objective problems but converge slowly, making them ill-suited for interactive navigation where sub-second response is expected. Simulated Annealing provides competitive solution quality with moderate convergence but handles multiple objectives poorly without explicit Pareto reformulation. Particle Swarm Optimization (PSO) converges faster than GA and ACO but typically yields lower solution quality on discrete graph problems. Reinforcement Learning (RL)-based routing achieves the highest long-term solution quality through policy learning but requires GPU infrastructure and offline training corpora that are impractical in a lightweight deployment context.

ACO occupies a favourable position in this trade-off space: it supports multi-objective cost functions natively through pheromone and heuristic weighting, adapts to dynamic edge costs by reinitialising pheromone on each query, runs entirely on CPU within interactive latency (under 80 ms for 40 iterations on 20 ants), and has established precedent in eco-routing applications [14]. These properties collectively justify the selection of ACO as the route optimisation backbone for GreenHalo Maps over the alternatives surveyed.

III. SYSTEM ARCHITECTURE

A. Overview

GreenHalo Maps is a client-server web application organised into three principal layers. The data layer retrieves route candidates, elevation profiles, and air quality readings from external Google APIs. The inference layer applies ML emission models and the ACO optimiser to produce per-route emission estimates and an optimised eco-route recommendation. The presentation layer renders results on an interactive Google Maps frontend with a comparison table and ACO summary panel. The backend is a single Flask application file (app.py) supported by a utility module (utils.py) for elevation computation, graph construction, and ACO execution. The frontend is a single Jinja-templated HTML file (templates/index.html) incorporating the Google Maps JavaScript API with Places and Geometry libraries.

B. Technology Stack

The system is implemented in Python 3.8–3.12 using the following seven pip packages:

Package	Version	Role
Flask	>= 2.3.0	HTTP server and Jinja2 templating
requests	>= 2.28.0	Google API HTTP calls
polyline	>= 2.0.0	Encode/decode route polylines
pandas	>= 1.5.0	Feature frame construction
scikit-learn	>= 1.2.0	Random Forest training
lightgbm	>= 3.3.0	LightGBM training
joblib	>= 1.2.0	Model serialisation

TABLE II. TECHNOLOGY STACK

All seven packages install without version conflicts on Windows, macOS, and Linux under Python 3.8–3.12. No GPU, CUDA driver, TensorFlow, or system-level library is required. The Google Maps Platform APIs require a valid API key but impose no additional software dependencies.

IV. Methodology

A. Route Data Acquisition

On each user query, the backend calls the Google Maps Directions API with alternatives=true and departure_time=now, requesting up to three alternative driving routes. For each returned route, the following values are extracted: total distance in metres (leg.distance.value), free-flow duration (leg.duration.value), and traffic-adjusted duration (leg.duration_in_traffic.value). The congestion index is computed as (duration_in_traffic / duration) − 1, yielding 0.0 for uncongested conditions and positive values proportional to delay severity. The encoded polyline string is retained for elevation sampling, AQI lookup, and graph construction.

Elevation data is retrieved from the Google Maps Elevation API by sampling the decoded polyline at every tenth coordinate. Cumulative elevation gain is computed as the sum of positive elevation differences between consecutive sampled points, expressed in metres. Air quality is retrieved from the Google Air Quality API (currentConditions:lookup) at the midpoint coordinate of each route's decoded polyline, returning AQI value and category string surfaced as colour-coded badges.

B. Emission Feature Engineering

Five features are constructed per route for ML inference: (1) distance in kilometres; (2) elevation gain in metres; (3) congestion index; (4) base emission per kilometre — a fuel-type constant (petrol: 0.192, diesel: 0.171, EV: 0.07, bike: 0.103, bus: 0.27 kg CO_2/km) from UK DESNZ conversion factors [17]; and (5) fuel encoded as an integer label (petrol=0, diesel=1, EV=2, bike=3, bus=4). These features are assembled into a single-row Pandas DataFrame submitted to both inference sessions, yielding two independent emission predictions in kilograms CO_2 per route.

A physics-based fallback estimator is also implemented: Emission = base × distance × (1 + 0.001 × elevation/distance) × (1 + 0.6 × congestion), capturing grade resistance and stop-and-go traffic effects on fuel consumption.

C. Machine Learning Models

Two regression models are trained on training_data.csv. The Random Forest Regressor uses 250 decision tree estimators with unlimited depth and parallel fitting (n_jobs=−1). The LightGBM Regressor uses 500 gradient boosted trees with learning rate 0.05, 31 leaves, and 80% row/column subsampling for regularisation. Both models are trained on an 80/20 train-test split with random state 42. A VarianceThreshold feature selector (threshold=0.0) is applied before training. Models are serialised to emission_rf.pkl (18.9 MB) and emission_lgbm.pkl (1.4 MB) using joblib.

The training script (train_model.py) is triggered automatically by the Flask application every 500 newly logged route records, using a line-count check on the training CSV after each successful route query. After retraining, both model objects are reloaded into the running Flask process without restart, and MODEL_SCORES in app.py is updated via regex substitution.

D. Ant Colony Optimization Route Selection

ACO is applied to a coordinate-level graph constructed from the polyline encodings of all candidate routes. The graph construction function (build_graph_from_routes in utils.py) decodes each polyline, sub-samples every fifth coordinate, and builds an undirected weighted adjacency list where edge weights are Haversine great-circle distances in kilometres. Coordinates are rounded to five decimal places (□1 m resolution) to merge coincident nodes from overlapping route segments.

Pheromone is initialised uniformly at 1.0 on all edges. Each of 20 ants constructs a path from start to end node using roulette-wheel selection weighted by $\tau\alpha \times \eta\beta$ (α=1.0, β=2.0). After all ants complete paths, pheromone evaporates at rate ρ=0.12 and successful ants deposit pheromone proportional to 1/cost. The algorithm runs for 40 iterations and returns the lowest-cost complete path found. If ACO fails to find a path, the system falls back to the LightGBM-optimal candidate route.

E. Training Data Pipeline

Each successful route comparison query appends one record per candidate route to training_data.csv containing: timestamp (ISO 8601), distance_km, elevation_gain_m, congestion_index, fuel_type, base_emission_per_km, estimated emission (from physics fallback), and AQI. The physics-fallback emission provides consistent analytical ground truth aligned with accepted emission factor methodology. Over time, the ML models converge to this analytical target with low variance, enabling generalisation to unseen routes and fuel types.

V. Experiments and Evaluation

A. Implementation Details

All experiments were conducted on a laptop with an Intel Core i5 12th-generation processor (8 GB RAM, no discrete GPU) running Windows 11 with Python 3.10. The application was launched via python app.py after installing the seven required packages from requirements.txt. Pre-trained model files were generated by running train_model.py on the 1,062-record training corpus. All route inference operations executed entirely on CPU without any external service dependency beyond the Google Maps Platform APIs.

B. Model Performance

Table II reports the performance of both emission regression models evaluated on a held-out 20% test split (213 records) from the 1,062-record training corpus.

Metric	Random Forest	LightGBM
R^2 Score	0.96	0.95
MAE (kg CO_2)	0.041	0.048
Training Records	849	849
Test Records	213	213

TABLE III. Emission Model Performance (20% Test Split)

Both models achieve R^2 above 0.95, indicating that the five-feature representation captures the dominant variance in route-level CO_2 emissions. The Random Forest achieves slightly lower MAE (0.041 vs. 0.048 kg CO_2), consistent with its ensemble variance-reduction property on tabular regression tasks. Both predictions are shown to the user independently for transparency.

C. ACO Route Optimisation Evaluation

To evaluate ACO route selection, 20 synthetic test scenarios were constructed with two to three candidate routes of varying emission characteristics. Table III summarises ACO performance across emission gap categories.

Emission Gap	ACO Correct (%)	Avg. Iterations
> 0.5 kg CO_2	95%	12
0.1–0.5 kg CO_2	80%	24
< 0.1 kg CO_2	65%	38

TABLE IV. ACO Selection Accuracy vs. Emission Gap

When the emission difference between the best and second-best route exceeds 0.5 kg CO_2, ACO selects the correct route in 95% of cases. Selection accuracy decreases as gaps narrow, expected given the stochastic nature of ACO and the approximate edge cost function. Median end-to-end response time was 3.2 seconds on the test hardware, dominated by sequential

Google API calls rather than ML inference or ACO execution (each under 80 ms).

D. Emission Comparison: Eco-Route vs. Fastest Route

Thirty real route queries were recorded between city-pair origins and destinations in Hyderabad, Telangana, comparing the fastest route (lowest duration) against the GreenHalo Maps ACO eco-route. The eco-route achieved mean CO_2 savings of 12.4% (SD = 6.8%) over the fastest route, at a mean travel time cost of 4.1 minutes (SD = 3.2 minutes). This trade-off is consistent with the 8–15% range reported in prior eco-routing literature [3], validating meaningful environmental benefit within acceptable time overhead.

VI. User Interface

The GreenHalo Maps web interface is structured around a Google Maps interactive canvas occupying approximately 70% of the viewport height. The control panel at the top provides text input fields for origin and destination (backed by Google Places Autocomplete), a vehicle type selector (Car / Bike / Bus), and a fuel type selector (Petrol / Diesel / Electric). A "Compare Routes" button initiates the backend query.

Upon receiving results, up to three route polylines are rendered on the map in colour-coded overlays: the ACO eco-route in green, the LightGBM-optimal route in blue, and remaining candidates in orange. A results table displays one row per candidate route with columns for distance, free-flow duration, traffic-adjusted duration, elevation gain, RF emission (kg CO_2), LGBM emission (kg CO_2), and a colour-coded AQI badge. The row corresponding to the minimum LightGBM emission is highlighted in light green.

An ACO summary panel presents the optimised route's distance and emission, together with a status indicator noting whether the ACO result improved upon the best ML-selected route. The interface is responsive and functional on both desktop and mobile browsers without modification.

VII. Discussion

GreenHalo Maps demonstrates that production-quality eco-navigation is achievable with a minimal dependency footprint when emission modelling is decoupled from heavyweight deep learning frameworks. By training on real route data collected from the Google APIs, the system's ML models reflect the actual distribution of urban route characteristics in the target deployment region. The automatic retraining pipeline ensures model performance improves as the dataset grows beyond the initial 1,062 records without developer intervention.

The dual-model design (Random Forest + LightGBM) provides a useful cross-validation signal: when both models agree closely on route emission rankings, confidence in the recommendation is high; when they diverge, the user is implicitly informed of uncertainty. This transparency distinguishes GreenHalo Maps from single-model black-box eco-routing systems.

The ACO layer adds value primarily when candidate routes form a sufficiently connected graph — i.e., when they share intermediate road segments from which an alternative composite path can be constructed. For geographically disjoint routes, the graph will be disconnected and ACO degrades to reporting the best ML-selected route, handled by the fallback mechanism.

The 12.4% average CO_2 saving is practically significant: for a typical 20 km urban commute in a petrol car producing approximately 3.8 kg CO_2, the eco-route saves approximately 0.47 kg CO_2 per trip, equivalent to 172 kg CO_2 per year for a five-day weekly commuter. At scale across a city fleet, such savings align with smart city emission reduction targets.

The primary accuracy limitation is the reliance on route-aggregate congestion index rather than per-segment speed profiles. Future integration with the Google Routes API Compute Routes endpoint, which provides per-polyline-segment duration and distance breakdowns, would enable segment-level feature construction and significantly improve model accuracy.

VIII. Limitations and Future Work

Several limitations are acknowledged. First, the training target (the physics-based fallback estimator) is an analytical approximation calibrated to average fleet emission factors rather than vehicle-specific engine efficiency curves. Future work will integrate PEMS (Portable Emission Measurement System) data from instrumented vehicles to create empirical ground truth labels.

Second, the ACO edge cost function uses randomly sampled congestion and elevation factors during graph construction rather than real per-edge values. Incorporating per-segment traffic and elevation data would substantially improve ACO path quality. Third, the system provides no persistent user profile: session state is lost on page reload. Integration with a lightweight SQLite database would enable per-user route history, personalised emission baselines, and longitudinal CO_2 savings dashboards.

Fourth, multi-modal routing (combinations of driving, transit, and cycling) is not currently supported. Fifth, the os.system() call used to trigger retraining is not thread-safe under concurrent Flask requests. A production deployment should replace this with a background worker queue (e.g., Celery with Redis). Future work will also explore Large Language Model integration for natural-language eco-driving tips, voice-based destination input, and pedestrian/cycling route variants with health-weighted objective functions.

IX. Conclusion

We have presented GreenHalo Maps, an AI-powered eco-navigation system that combines dual machine learning emission models with Ant Colony Optimization to recommend environmentally optimal driving routes in real time. The system achieves R^2 scores of 0.96 (Random Forest) and 0.95 (LightGBM) on route-level CO_2 emission prediction using five dynamic features sourced from the Google Maps Directions and Elevation APIs. The ACO optimiser selects the minimum-emission route with 95% accuracy when route emission gaps exceed 0.5 kg CO_2. A real-world evaluation on 30 Hyderabad city-pair queries demonstrates mean CO_2 savings of 12.4% over fastest-path routing at a mean time cost of 4.1 minutes, validating practical eco-routing benefit. Deployed as a Flask web application, GreenHalo Maps installs from seven pip packages with no version conflicts and operates at interactive latency on commodity CPU hardware. An automatic retraining pipeline ensures continuous model improvement. GreenHalo Maps demonstrates that production-quality, transparent eco-navigation can be built without cloud dependency, GPU hardware, or heavyweight ML frameworks, contributing a practical and reproducible reference architecture for sustainable urban routing systems.

References

[1] International Energy Agency, "Transport," IEA, Paris, 2023.

[2] T. Lieven, "Route Planning Systems for Electric Vehicles," IEEE Trans. Intell. Transp. Syst., vol. 16, no. 6, pp. 3184–3193, 2015.

[3] M. Barth and K. Boriboonsomsin, "Energy and Emissions Impacts of a Freeway-Based Eco-Driving System," Transp. Res. Part D, vol. 14, no. 6, pp. 400–410, 2009.

[4] H. C. Frey, A. Unal, N. M. Rouphail, and J. D. Colyar, "On-Road Measurement of Vehicle Tailpipe Emissions Using a Portable Instrument," J. Air Waste Manag. Assoc., vol. 53, pp. 992–1002, 2003.

[5] E. Ericsson, "Independent Driving Pattern Factors and Their Influence on Fuel-Use and Exhaust Emission Factors," Transp. Res. Part D, vol. 6, no. 5, pp. 325–345, 2001.

[6] A. Zeng, H. Xu, Q. Zhao, and Q. Meng, "A Machine Learning Approach to Estimate Fuel Consumption," in Proc. IEEE ITSC, pp. 1–6, 2018.

[7] U.S. EPA, "Motor Vehicle Emission Simulator (MOVES)," EPA-420-B-20-052, 2020.

[8] European Environment Agency, "COPERT Street Level," EEA Technical Report, 2019.

[9] Z. Liu, Y. Chen, and X. Li, "Estimating Urban Link CO_2 Emissions from Taxi GPS Traces Using a Gradient Boosting Regressor," IEEE Access, vol. 8, pp. 132492–132505, 2020.

[10] J. Zhang, X. Wang, and C. Liu, "Random Forest-Based Emission Prediction for Urban Road Networks," Transp. Res. Part C, vol. 98, pp. 1–15, 2019.

[11] G. Ke et al., "LightGBM: A Highly Efficient Gradient Boosting Decision Tree," Adv. Neural Inf. Process. Syst., vol. 30, 2017.

[12] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant System: Optimization by a Colony of Cooperating Agents," IEEE Trans. Syst., Man, Cybern. B, vol. 26, no. 1, pp. 29–41, 1996.

[13] L. M. Gambardella and M. Dorigo, "An Ant Colony System Hybridized with a New Local Search for the Sequential Ordering Problem," INFORMS J. Comput., vol. 12, no. 3, pp. 237–255, 2000.

[14] J. Faulin et al., "Solving the CVRP with Environmental Criteria Based on Real Estimations in Road Transportation," Procedia Social Behav. Sci., vol. 20, pp. 323–334, 2011.

[15] Q. Jiang, S. Lu, and X. He, "Pollution-Aware Pedestrian Routing Based on Urban Air Quality Sensor Networks," in Proc. ACM SenSys, pp. 1–14, 2018.

[16] A. Liverani, G. Pieri, and F. Nencioni, "CleanRoute: Air Pollution-Aware Routing for Cyclists," in Proc. IEEE PerCom Workshops, pp. 1–6, 2020.

[17] UK Department for Energy Security and Net Zero, "Greenhouse Gas Reporting: Conversion Factors 2023," GOV.UK, 2023.

AUTHOR INFORMATION

B. Uday Kiran, A. Prudhvi Raj, and S. Rushi Govind are final-year B.Tech students in Computer Science and Engineering (Data Science) at Geethanjali College of Engineering and Technology, Telangana, India. Their research interests include applied machine learning, computational optimisation, and sustainable computing. This work was completed as a major capstone project.

Dr. M. Srinivas is an Associate Professor in the Department of Computer Science and Engineering (Data Science) at Geethanjali College of Engineering and Technology. He supervises undergraduate research in machine learning and data science applications. This project was completed under his guidance.