

This is the peer reviewed version of the following article:

RI-PIENO - Revised and Improved Petrol-Filling Itinerary Estimation and Optimization / Savarese, Marco; De Blasi, Antonio; Zaccagnino, Carmine; Salici, Giacomo; Cascianelli, Silvia; Vezzani, Roberto; Grazia, Carlo Augusto. - (2025). (IEEE Consumer Communications & Networking Conference 2026 Las Vegas, NV, United States 09/01/2026) [10.1109/CCNC65079.2026.11366496].

Terms of use:

The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

29/04/2026 03:37

(Article begins on next page)

RI-PIENO – Revised and Improved Petrol-filling Itinerary Estimation aNd Optimization

Marco Savarese^{1,2}, Antonio De Blasi¹, Carmine Zaccagnino^{1,2}, Giacomo Salici^{1,2},
Silvia Cascianelli², Roberto Vezzani², Carlo Augusto Grazia²

Department of Engineering “Enzo Ferrari”, University of Modena and Reggio Emilia

¹{name.surname}@pieno.dev, ²{name.surname}@unimore.it

Abstract—Efficient energy provisioning is a fundamental requirement for modern transportation systems, making refueling path optimization a critical challenge. Existing solutions often focus either on inter-vehicle communication or intra-vehicle monitoring, leveraging Intelligent Transportation Systems, Digital Twins, and Software-Defined Internet of Vehicles with Cloud/Fog/Edge infrastructures. However, integrated frameworks that adapt dynamically to driver mobility patterns are still underdeveloped. Building on our previous PIENO framework, we present RI-PIENO (Revised and Improved Petrol-filling Itinerary Estimation aNd Optimization), a system that combines intra-vehicle sensor data with external geospatial and fuel price information, processed via IoT-enabled Cloud/Fog services. RI-PIENO models refueling as a dynamic, time-evolving directed acyclic graph that reflects both habitual daily trips and real-time vehicular inputs, transforming the system from a static recommendation tool into a continuously adaptive decision engine. We validate RI-PIENO in a daily-commute use case through realistic multi-driver, multi-week simulations, showing that it achieves significant cost savings and more efficient routing compared to previous approaches. The framework is designed to leverage emerging roadside infrastructure and V2X communication, supporting scalable deployment within next-generation IoT and vehicular networking ecosystems.

Index Terms—Intelligent Transportation Systems, IoT, Refueling Path Optimization, Time Series Analysis, V2X

I. INTRODUCTION

Wherever transportation of people or goods is required, the supply of energy becomes a fundamental constraint. This has motivated decades of research on the optimization of refueling paths, with approaches ranging from network-level routing optimization to AI-assisted driving strategies. These include the gas station problem [8], which has also been tackled with Reinforcement Learning [12]. On the networking side, Software-Defined Internet of Vehicles has been applied to QoS-aware routing [4], RSU cloud services [14], and heterogeneous multi-radio access management [7], while delay-tolerant schemes have been explored in cloud-fog vehicular systems [18]. Other contributions address time-sensitive in-vehicle networks [10] and cooperative paradigms such as platooning [5], [6]. Meanwhile, AI-based methods have been proposed to optimize energy consumption by learning from driver behavior [17]. Despite these advances, most works treat either inter-vehicle communication or intra-vehicle monitoring in isolation, and only rarely provide an integrated solution that is both practically deployable and adaptable across diverse mobility scenarios.

In our earlier work, *Petrol-Filling Itinerary Estimation aNd Optimization* (PIENO) [15], we presented a proof-of-concept framework combining intra-vehicle data (e.g., fuel level through CAN Bus and OBD-II [9], [13], [16]) with external data on fuel prices and petrol station locations, processed via cloud services. PIENO demonstrated how Intelligent Transportation Systems and Digital Twin concepts [3] can be combined to deliver personalized, eco-aware refueling recommendations. However, PIENO was conceived as a static framework: it focused on optimizing the next refueling event, employed simplified behavioral heuristics, and lacked mechanisms to adapt dynamically to recurring travel routines or longer-term usage contexts.

In this paper, we present a revised and improved version of PIENO, dubbed RI-PIENO, which addresses the aforementioned limitations. Specifically, RI-PIENO extends the PIENO framework by introducing:

- Temporal modeling of a driver’s habitual daily trips;
- Integration of machine learning predictions of user habits into Operations Research-based optimization;
- Representation of the refueling problem as a directed acyclic graph that evolves in time based on real-time sensor and geospatial inputs.

The combination of the original PIENO features and these novel contributions transforms the system from a one-off decision support tool into a continuously adaptive engine for refueling path optimization. Furthermore, the framework is designed to leverage emerging roadside infrastructure and V2X communication, paving the way for scalable, large-scale deployment scenarios.

We present RI-PIENO in the use case of personalized daily commutes and show how to mathematically model this scenario as a directed acyclic graph, built over time based on real data from the car’s sensors and geospatial information. Note that, although beyond the scope of this paper, the same formalism we present is straightforward to apply to the use case of single-day special journeys, which can be modeled as a simpler, shorter graph. We validate RI-PIENO through realistic simulations involving multiple drivers and paths over several weeks. Our results demonstrate that RI-PIENO outperforms both PIENO and baseline strategies, both in terms of cost and time saved, and also has a direct impact on reducing the user’s emissions by optimizing their refueling itinerary.

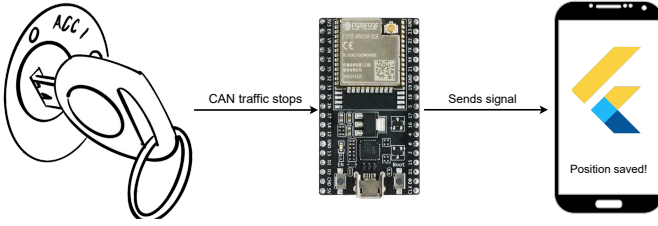


Fig. 1: Communication between RI-PIENO components.

II. METHOD

In this section, we present the main modules of the RI-PIENO framework, specifically designed to overcome the main limitation of PIENO [15], namely its lack of integration of user mobility patterns into the optimization process:

- A. Daily Trip Graph:** constructs a directed graph for each day of the week based on the user's identified *Points of Interest* (POIs). Each graph captures the principal daily movements and recurrent trip sequences.
- B. Weekly Mileage Prediction:** applies a machine learning approach to forecast the number of kilometers the user is expected to drive in the upcoming week. The prediction exploits historical trip data and an ensemble strategy capturing both non-linear patterns and overall trends.
- C. Fuel Stop Optimization:** leverages the information encoded in the daily trip graph together with an open-source routing engine to compute the optimal one-stop refueling route. Unlike PIENO, the stop is selected in coherence with the user's habitual trajectories.

A. Daily Trip Graph.

The *Daily Trip Graph* extraction module is in charge of modeling user-specific mobility patterns and information about the usual routes, which will be used by the *Fuel Stop Optimization* module. Specifically, this module processes raw location data, identifies the POIs, and reconstructs daily flow sequences in the form of directed graphs, whose nodes are the user's POIs and edges represent the usual route taken by the user to move from one POI to another. The graph is built on the basis of data collected from the vehicle during an observation period (in this work, we consider at least two weeks). The rationale behind this approach is to capture the typical flow of movements between usual destinations on a per-day basis, allowing the system to distinguish between regular mobility patterns (e.g., *home* \rightarrow *work* \rightarrow *gym*) and incidental stops, thereby enriching the system with personalized mobility knowledge. The module is organized into two main components:

- 1) **POI Creator**, responsible for clustering raw GPS coordinates into POIs based on spatial and temporal criteria;
- 2) **Flow Generator**, responsible for reconstructing daily trip flows by linking POIs into chronological sequences.

Both the POI Creator and the Flow Generator rely on a common base data structure that integrates information from the CAN Bus and the ESP32 [2], part of the CAM of PIENO [15],

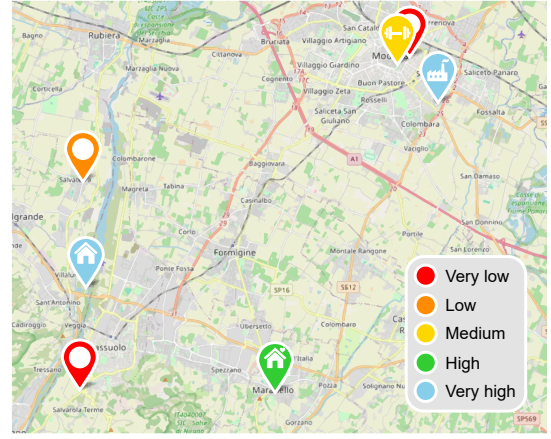


Fig. 2: Geographical map of all the clustered stops that a user has performed during the observation period, colored based on their frequency of visit that week. Note that only stops visited with at least Medium frequency are considered as POIs.

and is stored in the PIENO mobile application [15]. The specifics of this process will be described in Section II-A1 while their role and their output are detailed in the remainder of this subsection.

1) **POI Creator:** To build the daily trip graph, we first detect the vehicle stops through the CAN Bus. Specifically, when the car comes to a halt, the message flow on the CAN Bus is interrupted. Since these two events are strongly correlated, the absence of messages can be used as a proxy for vehicle stop detection. Once a stop is detected, the ESP32 sends a signal to the PIENO app, and the corresponding GPS position is stored and enriched with contextual metadata such as time, date, and timestamp. This data flow is depicted in Fig. 1. Note that, to prevent storing multiple stops all referring to the same location, we cluster new stops with previously recorded ones if they fall within a predefined radius (set to 100m in this study). This design choice also models the scenario in which the user parks in different spots close to their actual destination.

Each stop is a candidate POI node and is associated with a rich set of metadata:

- **Identifier:** Unique node label (e.g., STOP_001).
- **Coordinates:** Latitude and longitude of the centroid of the cluster to which nearby stops have been associated. These coordinates are obtained by combining those recorded by the PIENO app from the smartphone's GPS at the stops in the cluster.
- **Visit frequency:** Total number of visits to that location registered during the observation period, disaggregated into weekdays and weekends to enable refined analyses (even if beyond the scope of this work).
- **Frequency category:** Automatically assigned label based on the average number of visits to that node in a week (v). Specifically, the categories are VERY_LOW ($v \leq 1$), LOW ($1 < v \leq 2$), MEDIUM ($2 < v \leq 4$), HIGH ($4 < v < 10$), and VERY_HIGH ($v \geq 10$).

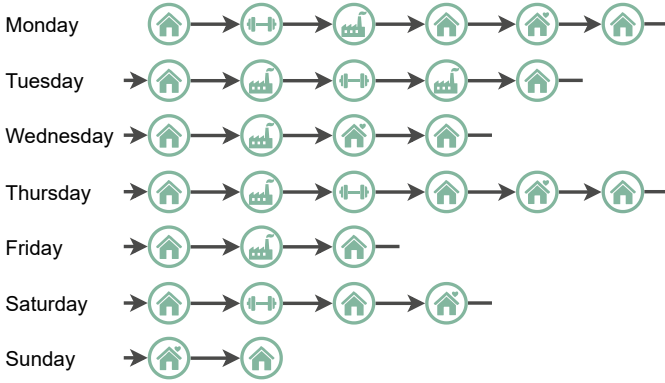


Fig. 3: Daily Trip Graph obtained after the user’s observation period, which reflects their typical weekly trips (*i.e.*, between ordered POIs) day by day.

- **Days visited:** Set of days in which the POI has been visited during the observation period (*e.g.*, Mon|Tue|Wed). These are reconstructed from the date metadata registered by the PIENO app at corresponding halt events.

A stop is then considered a user POI if it has been assigned one of the Frequency category labels from a pre-defined set. In this work, we consider `VERY_HIGH`, `HIGH`, and `MEDIUM` as acceptable labels (see Fig. 2). This structured information allows the system to distinguish highly relevant POIs (*e.g.*, home, workplace) from less significant ones (*e.g.*, occasional shops).

2) *Flow Generator:* The edges of the Daily Trip Graph represent the directed transitions between POIs observed during a single day. These are obtained by sorting the POI nodes based on the timestamp metadata registered by the PIENO app at corresponding halt events, and then tracing a directed edge from POI_i to POI_j between consecutive POIs (see Fig. 3).

For each transition, we store metadata associated with its destination node, with the implicit assumption that the previous entry represents the source. The stored information for each edge includes:

- **Day Identifier:** Tag that enables the optimizer (described in Section II-C) to restrict the analysis to the specific day under consideration.
- **Sequence index:** Integer that encodes the chronological order of the POI within the daily trajectory. This value is derived directly from the ordering of the stop events in the data structure of the application.
- **Coordinates:** Latitude and Longitude of the destination POI node.

From a practical point of view, the output of the Daily Trip Graph extraction module consists of two structured CSV files, one for the POI Nodes and one for the daily transitions (*i.e.*, the edges), containing their metadata. This modular design is straightforward to adapt and apply to novel scenarios and different applications (*e.g.*, it is easy to change the definition of POI Nodes, or to generate alternative transition hypotheses or routes for the same POIs). Therefore, although the scope of this work is fuel-stop recommendation, it is worth

noting that the devised daily trip graph provides a compact yet expressive representation of mobility patterns that can also support tasks like next-destination prediction, flow analysis, or personalized route optimization.

B. Weekly Mileage Prediction.

With the Daily Trip Graph, we obtain a picture of the consistent trips and the usual locations visited by a user over the week. Nonetheless, the user likely makes occasional trips to other locations. These are basically non-POI nodes. Therefore, in addition to their usual paths (and associated traveled distance, which can be obtained almost deterministically), the user will also travel to non-POI nodes every week. However, since these unusual locations are likely different from week to week, the path and traveled distance to reach them, which has an impact on the user’s fuel consumption, cannot be known a priori. To alleviate this issue, we devise a machine learning-based approach to estimate the mileage a user is likely to drive in the following week. Specifically, we employ a Random Forest [1] Regressor trained on historical trip data. The hyperparameters of the model are set to 150 estimators and a maximum depth of 6 after empirical analysis.

The predictor variables (features) capture both temporal patterns (such as the **day of the week** and the **month of the year**) and behavioral dynamics of travel (including **short-term lags** and **rolling averages of distance traveled**). In addition, when available, we include indicators of trip intensity and performance, such as the **number of trips**, and measures of **average and maximum speed**. Note that, to ensure comparability and improve model stability, we standardize all features prior to training.

We train the model on data collected during a variable number of weeks and validate it on an unseen (recent) week. To measure the model’s performance, we use the *mean absolute error (MAE)* computed between all the daily predictions, *i.e.*,

$$MAE = \frac{1}{N} \sum_{n=1}^N |y_n - \hat{y}_n|,$$

the *weekly absolute error*

$$E_{\text{week}} = \left| \sum_{n \in W} y_n - \sum_{n \in W} \hat{y}_n \right|,$$

and the *weekly percentage error*

$$E_{\text{week}\%} = \frac{E_{\text{week}}}{\sum_{n \in W} y_n} \times 100,$$

where y_n is the actual daily distance, \hat{y}_n is the predicted daily distance, W denotes the set of days in the week, and N is the total number of days evaluated.

Note that this data-driven predictive approach works best for users with consistent habits, which is not always the case. Therefore, RI-PIENO applies the model to predict the user’s mileage for the following week only if the validation is successful (*i.e.*, each prediction error is lower than the predefined threshold set as: $MAE \leq 2.5$, $E_{\text{week}} \leq 5.7$, and $E_{\text{week}\%} \leq 21.3$, in this work).

C. Fuel Stop Optimization.

Similar to what is done in PIENO [15], we exploit the weekly forecast of the fuel costs in the user’s area to select the most convenient day of the week for refilling. This directly translates into a portion of the Daily Trip Graph described in Section II-A. For that day, we compute the expected traveled distance between the POIs, d_n , and use the estimated weekly mileage \hat{y}_n (if deemed sufficiently reliable) to obtain an estimation of the extra distance traveled to reach unusual locations as:

$$\delta_n = \hat{y}_n - d_n.$$

In the following, we drop the suffix n for brevity.

Given the graph for the selected day, we find potential fuel stops along the user’s path, which we call *Fuel Station Nodes* (PS Nodes). To this end, we first compute the optimal routing between the ordered POIs using the open-source GraphHopper¹ routing engine based on OpenStreetMap² data (nonetheless, note that RI-PIENO is independent of the routing engine and can be used directly also with other systems, e.g., OSRM). Then, as in PIENO [15], we leverage government-provided fuel station data to obtain an exhaustive list of fuel stations in the user’s area, for which we also collect location coordinates and fuel prices. Finally, we filter the fuel stations by using their latitude and longitude and only keeping fuel stations that fall within a range of 2 km from the optimal route between the POIs (for computational reasons, but this radius can be enlarged). We refer to this list of candidate stops as *PS*: These PS nodes are *candidate refueling stops*, among which the best one will be selected by exploiting our devised optimization algorithm. For each candidate stop, we compute (again by exploiting GraphHopper) all possible one-stop routes starting from the closest, preceding POI node along the Daily Trip Graph. The output we get from GraphHopper includes time t_k and distance l_k to travel to each fuel station PS_k and reach the remaining POIs for the day, which can entail a short detour if the fuel station is not exactly along the user’s path. The distance value is corrected to account for weekly mileage variations, as $\hat{l}_k = l_k + \delta$. From these values, we obtain the fuel and time cost associated with the PS nodes, which are the input to our algorithm. The details are given below.

Fuel Cost. The fuel cost associated with the k -th PS node is computed as

$$C_k = (f_{full} - f_0 + r \cdot \hat{l}_k) \cdot c_k,$$

where f_{full} is the tank capacity of the vehicle, f_0 is the fuel level taken from the CAN Bus at the start of the trip, r is an estimate of the fuel consumption rate of the vehicle, and the C_k is the cost of the fuel at the fuel station PS_k , derived from the government data.

Time Cost. We determine the time cost associated with stopping at a given fuel station PS_k as the sum of the driving

time estimated by GraphHopper t_k and a fixed estimation of refueling time Δ :

$$T_k = t_k + \Delta.$$

Note that, in this work, we consider Δ as a constant with the same value for all the fuel stations (set to 5 minutes, in this work). Therefore, it will not have an impact on the selection of the best one. However, it is worth noting that this mathematical model allows for straightforward future extension in which Δ can vary depending on additional information coming from the fuel station (e.g., the number of vehicles currently queuing at the station) or other users connected via the PIENO app.

Given all the candidate fuel stops and their associated information, we first distinguish between reachable and unreachable stops. Unreachable stops are those that cannot be reached with the current level of fuel f_0 due to their distance \hat{l}_k and the user’s fuel consumption rate r . The optimization cost of these stops is set to infinity. Then, we find the index k of the optimal fuel station among the candidate stops as the one that minimizes the overall cost function:

$$\mathcal{L}_k = \begin{cases} K_1 \cdot C_i + K_2 \cdot T_i & \text{if } r \cdot \hat{l}_k \leq f_0 \\ \infty & \text{if } r \cdot \hat{l}_k > f_0, \end{cases}$$

where the two constants K_1 and K_2 are used to model the user’s preference to save cost or time. Specifically, by tweaking these two constants, we hypothesize three possible *modes* of operation of the system:

- **Fuel-sensitive**, if the user is particularly attentive to saving on fuel cost, and does not mind small detours to reach the optimal fuel destination. In our experiments, we set $K_1 = 1$ and $K_2 = 0$ to model this scenario.
- **Time-sensitive**, if the user is particularly attentive to saving time, and does not mind paying more for fuel. In our experiments, we set $K_1 = 1$ and $K_2 = 10$ to model this scenario.
- **Balanced**, if the user aims at balancing the two components. This is the default mode of the RI-PIENO system presented in this paper, for which we set the constants as $K_1 = 1$ and $K_2 = 1$.

III. EXPERIMENTS AND RESULTS

In this section, we validate our proposed approach and its components on a public dataset and on a set of recordings collected for this work.

A. Weekly Mileage Prediction.

To evaluate the proposed machine learning-based approach for the weekly mileage prediction on an independent set of data, we apply it to the VED dataset [11], a large-scale collection of operational data from hundreds of vehicles in real-world conditions. This dataset contains measurements of energy consumption and dynamic movement information, including GPS coordinates and speed, sampled every 1-3 seconds. To use the VED dataset in our experiments, we need to obtain a ground truth estimate for the daily distance traveled by each driver in the dataset. To this end, we aggregate the

¹<https://graphhopper.com/>

²<https://www.openstreetmap.org>

Obs. Period	MAE	E_{week}	$E_{\text{week}\%}$
4 weeks	4.27 ± 1.62	10.09 ± 4.50	42.0 ± 28.0
6 weeks	3.86 ± 1.54	9.04 ± 4.08	38.0 ± 25.1
8 weeks	3.59 ± 1.47	8.52 ± 3.97	34.2 ± 21.3

TABLE I: Random Forest performance on the same week, when trained on data collected during preceding observation periods (Obs. Period) of different lengths.

raw measurements available into total daily usage per vehicle. Specifically, we multiply the average speed recorded by the elapsed time between sampled positions. Note that we decide not to use a GPS-dependent approach (*e.g.*, calculating the geodesic distance between consecutive GPS samples based on their coordinates) since GPS signals may not always be available in real-world scenarios. Nonetheless, we have empirically found that our approach and a GPS-based one yield nearly identical results (0.166 km of median difference in their estimations).

For model evaluation, we perform time-series cross-validation using a sliding-window approach, where the training set consists of samples collected during a fixed number of historical weeks and the prediction horizon is the following week. The size of the historical training window is varied to examine its effect on the predictive performance. At each iteration, both daily and weekly forecasts are computed, and the results are recorded for aggregated statistical analysis across all validation folds. Table I presents the results of the Random Forest model when trained on data from different window sizes. The reported values are averaged over all 45 vehicles that traveled more than 1000 km during the one-year period covered by the VED dataset. The results demonstrate that with sufficiently long observation periods, the weekly mileage estimation is increasingly precise. Moreover, the relatively high standard deviations across predictions indicate that some travel profiles are more predictable than others (as argued in Section II-B). By taking into account these results on the independent VED dataset, we set the acceptability thresholds for integrating the output of the Weekly Mileage Prediction module into the RI-PIENO optimization. Specifically, we use the 25th percentile of the results obtained by training on 6 weeks’ worth of data.

B. Comparison with PIENO.

To assess the benefit of the optimization strategy in RI-PIENO, we consider 15 recordings obtained from 3 different users on different paths and periods, and compare them against PIENO and a baseline that entails simply stopping at the fuel station that is the closest one to the current user’s location, regardless of the fuel cost at that station.

Note that, for fair comparison between RI-PIENO and the competitors, we consider the visited locations and traversed paths from the users’ recordings and disregard their actual refueling stops. From these sets of data, we simulate what would have happened if the users were following the refilling advice given by the considered approaches when departing

	Cost [€]	Time [min]
Baseline	44.53 ± 2.20	6.61 ± 1.01
PIENO	43.24 ± 2.09	7.10 ± 2.23
RI-PIENO	43.01 ± 2.13	4.98 ± 0.80

TABLE II: Comparison between our proposed RI-PIENO, and simpler approaches that do not take into account the user’s behavior (PIENO) and the fuel cost information (Baseline).

K_1	K_2	Cost [€]	Time [min]
1	0	42.64 ± 1.77	7.65 ± 4.46
10	1	42.72 ± 1.84	5.50 ± 0.49
1	1	43.01 ± 2.13	4.98 ± 0.80
1	10	43.67 ± 1.98	4.87 ± 0.76
0	1	44.38 ± 2.44	4.85 ± 0.49

TABLE III: Effect of the optimization constants on the RI-PIENO solution. The first two rows reflect the *Fuel-sensitive* mode, the third the *Balanced* mode, and the last two the *Time-sensitive* mode.

from one of their POIs. In this way, we remove any variability due to daily external factors (*e.g.*, traffic, driver conditions).

For each path in the recordings set, we take the cost and time of refilling obtained with these approaches and average the results over all the recordings in the set. We report the obtained mean and standard deviation in Table II. As it can be observed, RI-PIENO consistently leads to cost and time savings. This performance improvement can be attributed to the fact that, different from PIENO, RI-PIENO takes into account the path that the user will likely take from the current POI (thanks to the Daily Trip Graph that models their habits). This is also showcased in Fig. 4, where we report the fuel station stop suggested by the Baseline, PIENO, and RI-PIENO to a user departing from a POI and directed to another one.

C. Operation Modes.

The most relevant hyperparameters of RI-PIENO are the optimization constants K_1 and K_2 , which model the importance to be given to saving fuel cost and time, respectively. Recall that different values given to the constants lead to various operation modes: **Balanced** ($K_1 = K_2$), which is the default setting, **Fuel-sensitive** ($K_1 > K_2$), and **Time-sensitive** ($K_1 < K_2$), which can be made more or less aggressive by changing the difference between the two constants. In Table III, we report the results obtained when changing their values in terms of saved cost and time. The results confirm the flexibility of the RI-PIENO approach: in the **Fuel-sensitive mode** (first two rows), we maximize the cost savings (as low as 42.64 €) at the expense of higher travel time (up to 7.65 min). On the contrary, in the **Time-sensitive mode** (last two rows), refueling time is heavily reduced (around 4.85 min) at the expense of lowering cost efficiency (up to 44.38 €). Finally, the **Balanced mode** (third row) offers an intermediate solution that overcomes both the Baseline and PIENO strategies.

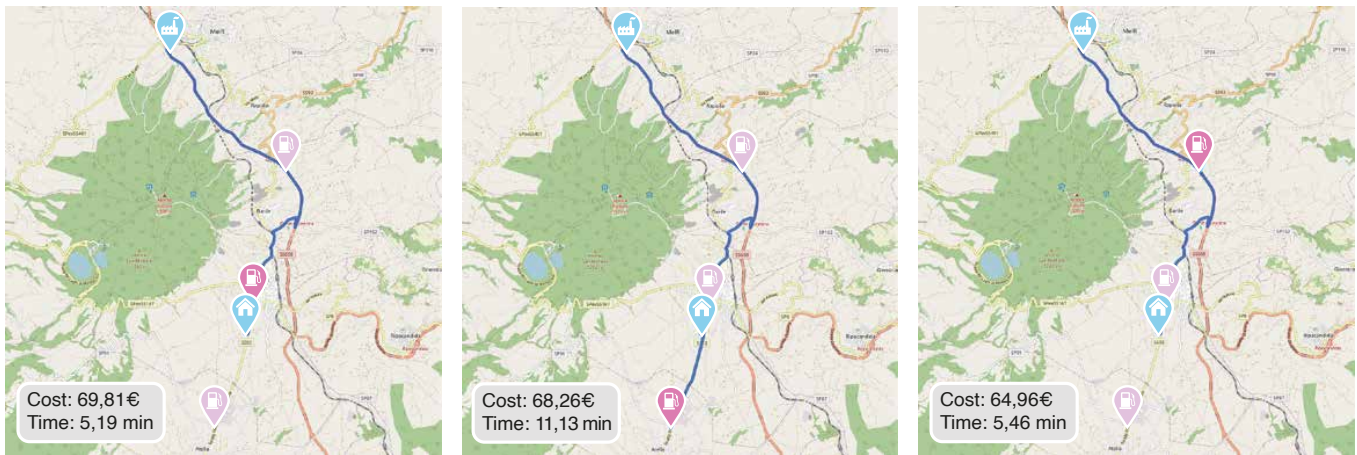


Fig. 4: User path from a POI (home) to another POI (workplace) and Fuel Station stop (dark pink) proposed by the Baseline (left), PIENO (center), and RI-PIENO (right), with relative cost and time. The fuel stations in the area are in light pink.

IV. CONCLUSIONS

In this paper, we have presented *RI-PIENO*, an improved and extended version of the PIENO framework for refueling path optimization. By integrating temporal modeling of daily trips, a time-evolving directed acyclic graph representation, and an Operations Research-based optimization module, *RI-PIENO* transforms the refueling recommendation process from a static decision tool into a continuously adaptive system. Another key strength of *RI-PIENO* lies in its flexibility. In fact, the system design makes use of several parameters, such as the fuel consumption rate r , the refueling time Δ , and the user preference weights K_1 and K_2 that balance cost and time. In this work, we have set these parameters to fixed, pre-defined values. Nonetheless, in future implementations, these parameters can be either refined through long-term observation of user behavior (e.g., empirical estimations of r from CAN Bus data) or explicitly set by the user through the application interface. This makes the framework straightforwardly extendable to a wide variety of driving styles, vehicle types, and user preferences. We have validated the efficacy of *RI-PIENO* through simulations based on real-world data collected from multiple users over several days. The results demonstrate significant savings and improved routing compared to both PIENO and baseline strategies.

ACKNOWLEDGMENT

This work was partially supported by MOST – Sustainable Mobility National Research Center and received funding from the European Union Next-GenerationEU (Piano Nazionale di Ripresa e Resilienza (PNRR) – Missione 4 Componente 2, Investimento 1.4 – D.D. 1033 17/06/2022, CN00000023).

REFERENCES

- [1] L. Breiman, “Random forests,” *Mach. Learn.*, vol. 45, no. 1, p. 5–32, 2001.
- [2] Espressif Systems, *ESP32 Series Datasheet*, 2023.
- [3] A. Fuller, Z. Fan, C. Day, and C. Barlow, “Digital Twin: Enabling Technologies, Challenges and Open Research,” *IEEE Access*, vol. 8, pp. 108 952–108 971, 2020.
- [4] K. Z. Ghafoor, L. Kong, D. B. Rawat, E. Hosseini, and A. S. Sadiq, “Quality of Service Aware Routing Protocol in Software-Defined Internet of Vehicles,” *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2817–2828, 2019.
- [5] F. Giannini, G. Franzè, F. Pupo, and G. Fortino, “A Sustainable Multi-Agent Routing Algorithm for Vehicle Platoons in Urban Networks,” *IEEE Trans. Intel. Transp. Syst.*, vol. 24, no. 12, pp. 14 830–14 840, 2023.
- [6] G. Guo and Q. Wang, “Fuel-Efficient En Route Speed Planning and Tracking Control of Truck Platoons,” *IEEE Trans. Intel. Transp. Syst.*, vol. 20, no. 8, pp. 3091–3103, 2019.
- [7] W. Huang, L. Ding, D. Meng, J.-N. Hwang, Y. Xu, and W. Zhang, “QoE-Based Resource Allocation for Heterogeneous Multi-Radio Communication in Software-Defined Vehicle Networks,” *IEEE Access*, vol. 6, pp. 3387–3399, 2018.
- [8] S. Khuller, A. Malekian, and J. Mestre, “To fill or not to fill: The gas station problem,” *ACM Trans. Algorithms*, vol. 7, no. 3, Jul. 2011.
- [9] Microchip Inc., *Stand-Alone CAN Controller with SPI Interface*, 2003.
- [10] S. Nam, H. Kim, and S.-G. Min, “Simplified Stream Reservation Protocol Over Software-Defined Networks for In-Vehicle Time-Sensitive Networking,” *IEEE Access*, vol. 9, pp. 84 700–84 711, 2021.
- [11] G. Oh, D. J. Leblanc, and H. Peng, “Vehicle Energy Dataset (VED), A Large-Scale Dataset for Vehicle Energy Consumption Research,” *IEEE Trans. Intel. Transp. Syst.*, vol. 23, no. 4, pp. 3302–3312, 2022.
- [12] A. L. C. Ottoni, E. G. Nepomuceno, M. S. d. Oliveira, and D. C. R. d. Oliveira, “Reinforcement learning for the traveling salesman problem with refueling,” *Complex Intell. Syst.*, vol. 8, no. 3, pp. 2001–2015, 2022.
- [13] SAE International, “E/E Diagnostic Test Modes,” SAE International, Tech. Rep. J1979_201702, 2017.
- [14] M. A. Salahuddin, A. Al-Fuqaha, and M. Guizani, “Software-Defined Networking for RSU Clouds in Support of the Internet of Vehicles,” *IEEE Internet Things J.*, vol. 2, no. 2, pp. 133–144, 2015.
- [15] M. Savarese, A. De Blasi, C. Zaccagnino, and C. A. Grazia, “P.I.E.N.O.–Petrol-Filling Itinerary Estimation and Optimization,” *IEEE Access*, vol. 12, pp. 158 094–158 102, 2024.
- [16] G. Signoretti, M. Silva, A. Dias, I. Silva, D. Silva, and P. Ferrari, “Performance Evaluation of an Edge OBD-II Device for Industry 4.0,” in *MetroInd4.0&IoT*, 2019, pp. 432–437.
- [17] H. Sun, F. Tao, Z. Fu, A. Gao, and L. Jiao, “Driving-Behavior-Aware Optimal Energy Management Strategy for Multi-Source Fuel Cell Hybrid Electric Vehicles Based on Adaptive Soft Deep-Reinforcement Learning,” *IEEE Trans. Intel. Transp. Syst.*, vol. 24, no. 4, pp. 4127–4146, 2023.
- [18] B. Xia, F. Kong, J. Zhou, X. Tang, and H. Gong, “A Delay-Tolerant Data Transmission Scheme for Internet of Vehicles Based on Software Defined Cloud-Fog Networks,” *IEEE Access*, vol. 8, pp. 65 911–65 922, 2020.