Expert Systems With Applications 317 (2026) 131856

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

A bi-objective sub-path location model for the flow refuelling location problem

Bruno Salezze Vieira [a,b,c,*], Glaydston Mattos Ribeiro [d], Antônio Augusto Chaves [b]

[a] Aeronautics Institute of Technology, São José dos Campos, Brazil
[b] Federal University of São Paulo, São José dos Campos, Brazil
[c] Southampton Business School, University of Southampton, Southampton, UK
[d] Transportation Engineering Program, Alberto Luiz Coimbra Institute Institute - Graduate School and Research in Engineering, Federal University of Rio de Janeiro, Rio de Janeiro, Brazil

ARTICLE INFO

Keywords:
Location problem
Alternative fuel vehicles
Bi-objective
Exact methods
Heuristics.

ABSTRACT

Strategic placement of alternative-fuel refuelling stations is a critical challenge for energy and transportation planners, who must navigate conflicting objectives, such as minimising capital costs and maximising service coverage. This paper presents a decision support system for the bi-objective Flow Refuelling Location Problem (FRLP) built upon the Sub-path Flow Refuelling Location Model (SPFRLM). This formulation distinguishes itself by enabling continuous facility siting along edges, managed through a dynamic separation procedure for sub-path constraints. To generate solutions efficiently, the system incorporates the Smoothest Descent Algorithm (SDA), a bi-objective method that approximates the Pareto front by dynamically switching between minimization and maximization strategies. The SDA relies on a Two-Phase Hybrid (TPH) algorithm that integrates Cut-and-Solve and Branch-and-Cut to solve the underlying sub-problems. We validate the system on a newly introduced library of 26 real-world test instances. The results demonstrate that the proposed approach captures approximately 97% of the optimal hyper-volume while requiring less than 10% of the computational time of exact methods. These findings confirm that the system is a powerful tool for stakeholders, providing rapid and accurate guidance for the strategic deployment of future energy infrastructure.

1. Introduction

The remarkable growth of alternative fuel vehicles (AFVs) in recent years has created an urgent need for expanded refuelling station (RS) infrastructure. While electric and hydrogen-fueled vehicles offer a promising pathway to decarbonize road transport and reduce dependence on fossil fuels (Anjos et al., 2020), their widespread adoption hinges critically on the availability of convenient refuelling options. Consequently, the strategic placement of this infrastructure has emerged as a critical management challenge for government and industry stakeholders, with the lack of stations identified as a primary barrier to market penetration (Yildiz et al., 2016).

Formally known as the Flow Refuelling Location Problem (FRLP), the problem of locating these facilities is not merely a logistical puzzle but a complex decision-making process involving significant economic risk. For instance, strategically placed stations—situated in high-density areas or along major highways—can accelerate adoption by enhancing convenience. Conversely, poorly sited facilities represent a waste of capital and can actively discourage potential buyers (Zhang et al., 2019). Navigating this complexity requires intelligent systems capable of providing expert guidance to address the following key challenges:

- **Data Scarcity:** In many regions, detailed data on travel patterns and demand for AFV refuelling is unavailable. Effective systems must therefore derive robust insights from limited, publicly accessible datasets.
- **Competing Objectives:** Decision-makers face a classic conflict between minimizing high capital investments and maximizing service coverage. Balancing these trade-offs is essential for sustainable project management.
- **Computational Complexity:** The sheer scale of real-world road networks precludes manual planning. Expert systems must leverage powerful algorithms to search the vast solution space efficiently.

This paper explicitly addresses the challenges of limited data availability and financial constraints. To overcome data scarcity, our approach relies on minimal inputs: a road network topology and an

* Corresponding author.
E-mail addresses: bsvieira@unifesp.br (B.S. Vieira), glaydston@pet.coppe.ufrj.br (G.M. Ribeiro), antonio.chaves@unifesp.br (A.A. Chaves).

https://doi.org/10.1016/j.eswa.2026.131856
Received 3 July 2025; Received in revised form 17 February 2026; Accepted 24 February 2026
Available online 1 March 2026

estimated Origin-Destination (OD) matrix. Both can be sourced from public repositories or derived using population-based gravitational models (Hodgson, 1990a). To address conflicting objectives, we present bi-objective optimal curves that quantify the trade-offs between coverage and installation costs. These visualizations provide decision-makers with the clear insights needed to balance service levels against investment limits.

This study advances the literature on the Flow Refuelling Location Problem (FRLP) through contributions in three primary areas: mathematical modeling, solution algorithms, and experimental validation.

First, regarding modeling, we propose the Sub-path Flow Refuelling Location Model (SPFRLM). Unlike traditional formulations that often restrict facilities to discrete nodes or require network expansion (Capar and Kuby, 2012; Capar et al., 2013; Kuby and Lim, 2005; MirHassani and Ebrazi, 2013; Wang and Lin, 2009; Yildiz et al., 2016), our approach enables continuous siting along edges. This is achieved through a set of sub-path constraints that ensure valid refuelling intervals along any candidate path. To address the combinatorial magnitude of these constraints, we introduce a separation process that validates solution feasibility and generates cuts dynamically without requiring full constraint enumeration.

Second, in the algorithmic aspect, we develop the Smoothest Descent Algorithm (SDA), a bi-objective method that efficiently approximates the Pareto front. The novelty of the SDA lies in its timeline structure and switching-trigger logic, which dynamically alternates between minimizing capital costs and maximizing coverage depending on the density of the solution space. This is complemented by a Two-Phase Hybrid (TPH) cover algorithm, which integrates the dual-bound efficiency of Cut-and-Solve with the primal precision of Branch-and-Cut to solve each underlying ϵ-constrained sub-problem effectively.

Third, regarding the experimental component, we introduce a new library of 26 large-scale, real-world instances derived from the Brazilian national road network. These instances significantly exceed the scale of many common benchmarks found in the FRLP literature and are made publicly available to foster reproducibility and future benchmarking. Through these experiments, we present a rigorous bi-objective analysis, demonstrating that the proposed system can capture the vast majority of the optimal Pareto curve performance-cost trade-off while requiring only a fraction of the computational time of an exact method.

This paper is organized as follows. Section 2 reviews the related literature. Section 3 formally defines the problem and the proposed mathematical model. Section 4 presents the solution methods with the SDA, formulation solution methods, and separation procedure. Section 5 follows the computational results and the accompanying discussion in Section 6. So, we conclude the study in Section 7.

2. Literature review

The Flow Refuelling Location Problem (FRLP) is a distinct category within the broader family of facility location problems (FLP), which boasts extensive literature (Church and Drezner, 2022; Farahani et al., 2019; Melo et al., 2009). Given the specificity and importance of refuelling infrastructure, a specialized body of work has emerged focusing on flow-capturing objectives (Kchaou-Boujelben, 2021).

The FRLP is a problem within a broader family of AFV refuelling station location problems. Although similar to this work, multiple authors have presented solutions from a decision-making perspective, mostly using the same input data. Most models for solving it are adapted from the flow-capturing location model, proposed by Hodgson (1990b), to alternative-fuel refuelling stations, given the limited vehicle range, and are referred to as the flow refuelling location model (FRLM). Starting with Kuby and Lim (2005), with pre-calculation of refuelling station combinations, Kuby and Lim (2007) studied multiple algorithms for the dispersion of candidate sites on arcs, and Wang and Lin (2009) added binary variables for fuel depletion simulation.

Later, Kuby et al. (2009) expanded the Kuby and Lim (2005) formulation to more cases and presented heuristics at Lim and Kuby (2010) to solve larger instances. The same formulation was extended in Kim and Kuby (2012) to yield flow deviation paths per OD trip, referred to as the deviation-flow refuelling location model (DFRLM), which provides more granular flow coverage per path but requires path enumeration. This work was extended in Kim and Kuby (2013), which introduced network transformation heuristics that significantly improved the ability to solve larger instances.

Followed up by Capar and Kuby (2012) and Capar et al. (2013), which proposed more efficient formulations that did not require refuelling station combination calculations per path. MirHassani and Ebrazi (2013) presented an FRLM that ensures complete coverage by adding artificial nodes and edges to create an extended network. This formulation was generalized by Chung and Kwon (2015) by adding multiple periods, and Arslan and Karaşan (2016) adapted it to hybrid vehicles and multiple vehicle ranges, presenting a series of valid inequalities and showing that it is more efficient than previous FRLMs. The location with flow deviations was also addressed in Yildiz et al. (2016), which presented three new formulations that improved prior models. The most efficient one, called PS, is based on column generation and solved with a branch-and-price algorithm. Lee and Han (2017) proposed a model to solve the problem with uncertainties and a Benders and Price method to obtain better solutions than the other two models, and use a commercial solver.

More recently, Arslan et al. (2019) proposed a *q-node-cuts*-based model (QNC-FRLM) with multiple branch-and-cut algorithms; their best branch-and-cut algorithm significantly outperformed prior solution methods. Göpfert and Bock (2019) introduced another cut-based formulation, inspired by the duality between max flow and min cut, for vehicles with limited driving ranges, and a branch-and-cut approach that, for the first time, enables the efficient solution of some large-scale instances to optimality. Finally, Kinay et al. (2021) presented a full-cover modelling framework for locating fast-charging stations for battery-electric vehicles, ensuring that all origin-destination trips are covered while optimizing routes and minimizing recharging needs. It proposed two model variants and significantly outperforms previous full-cover approaches.

Despite these advancements, a gap remains in the effective handling of continuous location decisions in large-scale networks without excessive preprocessing or network expansion. Most existing approaches rely on node-based restrictions or simplified assumptions about deviation that may not capture the granular reality of long-distance travel. The following section defines the specific problem environment and premises adopted in this study to address these limitations.

Other studies consider specific data and problem characteristics to locate electric RSs. Yang et al. (2017) presented a model that uses GPS data collected from the city's taxi fleet of Changsha, means of regression, and logarithmic transformations. Anjos et al. (2020) solved the problem in a multi-step deployment scenario using a rolling-horizon-based heuristic that obtained better solutions than the MIP solver. And there are also approaches that prioritize deploying RSs along designated corridors to better serve high-demand areas (Chen et al., 2016; Erdoğan et al., 2022).

Regarding the location of hydrogen RSs, Greene et al. (2020) reviewed the challenges of designing, planning, and deploying hydrogen refuelling infrastructure, and the life-cycle environmental effects of hydrogen vehicles, depending on how hydrogen is produced and delivered to refuelling stations. Kim et al. (2020) proposed a three-step approach, with one model per step, to develop a strategic deployment plan over 18 years for the South Korean case. Their approach accounts for demand estimates, each RS's capacity, and sequential deployment. Finally, Seo et al. (2020) studied the construction of a hydrogen supply chain model using a centralized storage system to efficiently supply the RSs in the South Korean case.

Among studies that located RSs with zone centroids for specific cities or metropolitan zones, we have Liu (2012) that used the city of Beijing as a case study, proposed an assignment model to locate the RSs, simulated

the impact of the new RSs on the current power grid, and studied battery swap strategies. Chen et al. (2013) assigned RSs to park spots with a model with site accessibility, average parking time and costs, local jobs, population densities, trip attributes, travel surveys, and several other data from Seattle. Dong et al. (2014) proposed an activity-based assessment model, power grid impacts, heterogeneous RSs with their costs and capacities, and GPS-based travel survey data collected in the Seattle metropolitan area to justify the infrastructure investment. Tu et al. (2016) presented a spatial-temporal demand coverage approach with massive GPS data from Shenzhen and analyzes the trade-off between minimizing RSs queues and the cost of RSs installed. A continuum-approximation model was proposed, and its deployment was simulated in Li et al. (2016). Zhu et al. (2016) proposed a region-based model for electric RSs and a genetic algorithm, and studies the deployment impacts in Beijing. More recently, Kınay et al. (2023) presented a stochastic model for capacitated charging stations using bi-level optimization, where the network planner or leader minimizes the total infrastructure cost while ensuring a probabilistic service requirement on the waiting time to charge.

Beyond the location-specific and methodological studies discussed above, the landscape of alternative fuel infrastructure is well documented in several comprehensive surveys. For more literature on hydrogen-fueled vehicles, we refer the reader to Lin et al. (2020). Similarly, studies for electric vehicle charging networks are thoroughly reviewed by Zhang et al. (2019), Majhi et al. (2021), and Kchaou-Boujelben (2021).

3. Problem definition and modelling

The Flow Refuelling Location Problem (FRLP) is defined as the optimization challenge of siting refuelling facilities to intercept vehicles traveling between Origin-Destination (OD) pairs, thereby maximizing total flow coverage (Arslan and Karaşan, 2016; Kim and Kuby, 2012; Yildiz et al., 2016). Addressing the FRLP is crucial, as the strategic availability of Refuelling Stations (RSs) is a primary determinant of widespread AFV adoption. Site selection must account for critical factors such as accessibility, proximity to demand centers, and the availability of power or hydrogen infrastructure (Lin et al., 2020; Majhi et al., 2021). To address these requirements, we present a formulation grounded in three key operational premises:

- **Boundary Charging:** We assume every origin and destination is equipped with charging capabilities. Consequently, all vehicles depart fully charged, and a trip is considered feasible if the vehicle can reach its destination or an intermediate RS before depletion. Accordingly, all RSs must be spaced at intervals no greater than the vehicle's range.
- **Bidirectional Service:** RSs are assumed to serve traffic in both directions of a road segment, implying a network model with undirected edges.
- **Path Deviation:** Drivers do not always choose the absolute shortest path; they may prefer routes offering better scenery, road quality, or lower tolls. To capture this behavior, our model considers all "deviation paths" within a specified threshold of the shortest route as potential trajectories for an OD pair.

A critical distinction of our work lies in how we treat these deviation paths. Most existing literature assumes that an OD pair is covered if *at least one* valid path is traversable. This "single-path" assumption implicitly requires drivers to identify and select the specific route with infrastructure, sacrificing convenience for lower initial investment costs. In contrast, our framework adopts a robust coverage criterion: an OD pair is considered covered only if *all* candidate paths within the deviation threshold are covered. This approach prioritizes driver convenience, ensuring that users can select any reasonable route without "range anxiety" or the need for strict route planning. While this may require a higher investment and is suitable for mature or well-funded deployments, it significantly enhances the network's robustness.

Formally, we define a *candidate path* as any path between an OD pair whose length does not exceed the minimum path length by more than a specified tolerance. A candidate path is considered covered if every segment (or *sub-path*) exceeding the vehicle's range contains at least one RS. Our objective is to simultaneously maximize the number of covered trips (coverage) and minimize the required number of RSs (cost), providing decision-makers with a comprehensive view of the trade-offs involved.

Fig. 1 illustrates a small road network with five OD nodes (a, b, c, d, e) with a Trip Table. To better understand this description of candidate paths and sub-paths, consider a maximum deviation of 20% more than the shortest path and a range of 30 for the AFVs. For example, OD pair db (or bd) has a shortest distance of 65, with path $\overline{dkjb}$, leading to a maximum deviated path between d and b of distance 78. Therefore, candidate path $\overline{dkfijb}$ has a length of 76 $(10 + 15 + 6 + 20 + 25)$, and it is one of the sub-paths that needs to be covered to cover db. Fig. 2 illustrates this path with the proportional distances of each edge. If an AFV uses this path, it must refuel twice to complete it. Consequently, RSs must be located along this path to ensure coverage. However, if both are located between d and k, it would leave portions of the path with lengths greater than 30 without any RS. Thus, the condition to cover paths only is too weak to solve the FRLP. Therefore, we must cover all sub-paths with lengths greater than 30, such as $\overline{kfij}$ (41) requires at least one RS, $\overline{ijb}$ (45) one, and $\overline{kfijb}$ (66) two RSs. A possible solution to cover this candidate path is to locate one RS in each edge $\overline{fi}$ and $\overline{jb}$.

Here, we introduce a new mathematical model, the Sub-path Flow Refuelling Location Model (SPFRLM), to solve the FRLP as a bi-objective problem, aiming to maximize coverage (total number of trips) and minimize costs (total number of RSs). An optimal bi-objective curve between cost and coverage clearly shows the trade-offs between the objectives, inflexion points, and zones of opportunity a given network may have, helping a decision-maker make more informed decisions. We also present the Smoothest Descent Algorithm (SDA), an ϵ-constraint-based algorithm to solve this model. For each ϵ-constrained case, we propose and evaluate multiple solution methods as an Enumerated Cover (EC), a Branch-and-cut (B&C), a Cut-and-solve (C&S), and a hybrid approach called Two-Phase Hybrid (TPH). The SPFRLM is easy to adapt to solve different variants of the FRLP.

3.1. Mathematical model

Let $G = (N, A)$ be a road network, where A is the set of bi-directional roads, and $N = T \bigcup I$ is the set of nodes. Set T represents OD nodes, and set I indicates all network intersections. Set $W = \{(o, d) \mid o, d \in T,\ o \neq d\}$ is the set of OD pairs. For each $w = (o, d) \in W$, t_w is the number of trips for w, set P_w denotes all candidate paths between origin o and destination d shorter than $D_w^{\max} := \delta \times D_w^{\min}$, with $D_w^{\min}$ being the shortest distance from o to d, and δ a proportion constant. Each candidate path p has a set of sub-paths $\mathcal{SP}_p$ containing all sub-paths longer than R, which is the vehicle's range. Formally, to cover any path or sub-path s, given that $|s|$ is the total length of s, the condition to cover s is to have at least K_s RSs located in s. Considering every trip starts with a fully charged vehicle, in this paper, K_s is calculated as $\left\lfloor \frac{|s|-\epsilon}{R} \right\rfloor$. Table 1 guides the reader with this notation with definitions, sets, attributes, parameters, and model variables.

Applying this notation to Fig. 1, this small road network is composed of $|A| = 23$ edges, $|N| = 17$ nodes, with $T = \{a, b, c, d, e\}$ as the OD nodes, and the remaining $|I| = 12$ are the intersection nodes. The OD pairs are $W = \{(a, b), (a, c), (a, d),\ (a, e), (b, c), (b, d), (b, e),\ (c, d), (c, e), (d, e)\}$, and there are several paths connecting them.

For example, looking at the OD pair (a, d) and let $\delta = 1.2$, $D_{ad}^{\min} = 50$ (from path $\overline{afd}$), $R = 30$, paths $\overline{afd}$ (50) and $\overline{afkd}$ (55) connect origin a to destination d belong to set P_{ad}, but $\overline{alqd}$ does not as $|\overline{alqd}| = 120$,

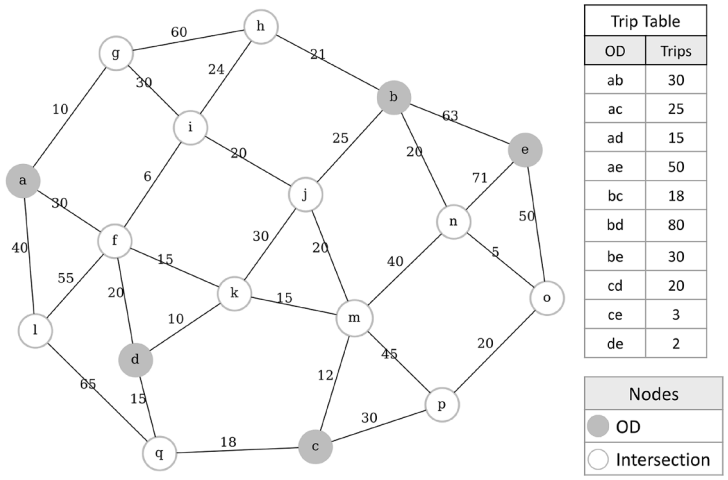

Fig. 1. Example of a road network with five OD nodes, 12 connectors, and an expected number of trips between OD pairs.

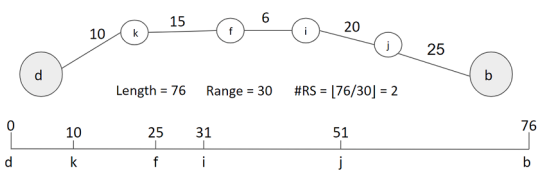

Fig. 2. Path coverage example.

Table 1
Notations.

Definitions	
Edge	Sorted pair of connected nodes with a length
Sub-path	Sorted set of connected nodes or edges
Path	Sub-path whose first and last edge contains OD nodes
$\|s\|$	Total length of sub-path or segment s
$\overline{abc}$	Connected segment from node a to b and c
Sets	
G	Network Graph
N	Nodes
A	Edges
T	OD nodes
I	Intersection nodes
W	OD pairs
S	Sub-paths
L	Continuous locations
P_w	Candidate paths for pair w
S_w	Sub-paths to cover pair w
W_S	OD pairs from the sub-paths of S
A_S	Edges from the sub-paths of S
Attributes	
t_w	Amount of trips for pair w
Γ	Total amount of trips
$D_w^{\max}$	Maximum distance a trip for pair w is likely to take
$D_w^{\min}$	Minimum distance trip for OD pair w
K_s	Amount of RSs necessary to cover sub-path s
$K_{A'W'}$	Amount of RSs on A' necessary to cover W'
$C^{\min}$	Minimum amount of RSs to cover all trips
$\|PS\|$	Number of calculated Pareto solutions
Parameters	
ϵ	Smallest positive number
δ	Maximum distance threshold
R	AFV range
tl_1	Time limit per Pareto point
tl_2	Time limit per Cut & Solve step
Model Variables	
x_a	Number of refuel stations on edge a
y_w	1 if OD pair w is covered, 0 otherwise

that is greater than $D_{ad}^{\max} = 1.2 \cdot 50 = 60$. Moreover, set S_{ad} contains $\overline{afd}$ and $\overline{afk}$ (from path $\overline{afkd}$), but none on $\overline{af}$, $\overline{fd}$ and $\overline{kd}$ since they are smaller or equal than the range R.

For the decision variables, $x_a \in \mathbb{Z}^+$ is an integer variable indicating how many RSs should be located on edge a ($x_a \geq 1$) or not ($x_a = 0$), and $y_w \in \{0, 1\}$ is a binary variable that indicates if OD pair w was covered ($y_w = 1$) or not ($y_w = 0$). In this paper, we propose a bi-objective model called the Sub-path Flow Refuelling Location Model (SPFRLM). The objective function (1) minimizes the number of RSs, and the objective function (2) maximizes the number of covered trips. Constraints (3) ensure that any sub-path if covered, has at least $K_s = \left\lfloor \frac{|s|-\epsilon}{R} \right\rfloor$ RSs.Constraints (4) ensure that any set of edges A' and set of OD pairs W', if covered, requires at least $K_{A'W'}$ RSs. Finally, Constraints (5) - (6) define the domain of the decision variables.

$$\text{Minimize } z_1 = \sum_{a \in A} x_a \tag{1}$$

$$\text{Maximize } z_2 = \sum_{w \in W} t_w y_w \tag{2}$$

Subject to:

$$\sum_{a \in s} x_a \geq K_s y_w \qquad \forall s \in S_w, w \in W \tag{3}$$

$$\sum_{a \in A'} x_a \geq K_{A'W'} \Big(\sum_{w \in W'} y_w - |W'| + 1 \Big) \qquad \forall A' \subset A, W' \subset W \tag{4}$$

$$x_a \in \mathbb{Z}^+ \qquad \forall a \in A \tag{5}$$

$$y_w \in \{0, 1\} \qquad \forall w \in W. \tag{6}$$

Additionally, Constraints (7) and (8) are used alternatively in conjunction with the opposing objective for the proposed ϵ-constraint method, where α is a positive integer that limits the number of RSs, usually described as a budget, $\beta \in [0, 1]$ states the minimal coverage level, and $\Gamma := \sum_{w \in W} t_w$ is a constant to normalize β.

$$\frac{1}{\Gamma} z_2 \geq \beta \tag{7}$$

$$z_1 \leq \alpha \tag{8}$$

One attribute of the SPFRLM (1) - (6) is the number of constraints (3) (sub-paths) that grow very fast depending on instance sizes, such as this drove our approaches to solve the problem, as detailed in the following subsection.

3.2. Constraints

Constraints (3) are our main set of constraints that can be naturally read. For example, taking the road network from Fig. 1 and the pair

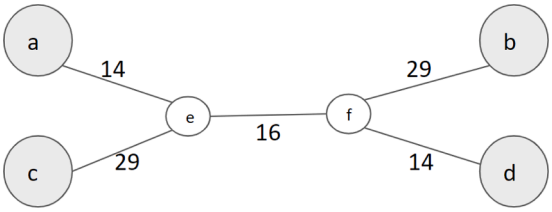

Fig. 3. Multi-Path location conflict example.

$w = (b, d)$, the shortest path is $\overline{bjkd}$ (65). Considering $\delta = 1.1$ and $R = 30$, we have four candidate paths ($P_w = \{\overline{bjkd}, \overline{bjmkd}, \overline{bjifd}, \overline{bhifd}\}$), with distances equal to 65, 70, 71, and 71, respectively. All other paths exceed the maximum distance threshold $D_{bd}^{\max} = \delta R = 71.5$. Therefore, these candidate paths require at least $K_p = \left\lfloor \frac{65}{30} \right\rfloor = \left\lfloor \frac{71}{30} \right\rfloor = \left\lfloor \frac{71}{30} \right\rfloor = 2$ RSs if OD pair (b, d) is covered. Considering just $p = \overline{bhifd}$, the sub-paths longer than R and shorter than $D_{bd}^{\max}$ are $\{\overline{bhi}, \overline{bhif}, \overline{bhifd}, \overline{hifd}\}$ with their respective lengths 45, 51, 71 and 60, more over, each respective K_s is $\left\lfloor \frac{45}{30} \right\rfloor = 1$, $\left\lfloor \frac{51}{30} \right\rfloor = 1$, $\left\lfloor \frac{71}{30} \right\rfloor = 2$ and $\left\lfloor \frac{60-\epsilon}{30} \right\rfloor = 1$. Hence the respective constraints (3) for pair $w = (b, d)$ and path $p = \overline{bhifd}$ are:

$$x_{\overline{bh}} + x_{\overline{hi}} \geq y_{bd}$$

$$x_{\overline{bh}} + x_{\overline{hi}} + x_{\overline{if}} \geq y_{bd}$$

$$x_{\overline{bh}} + x_{\overline{hi}} + x_{\overline{if}} + x_{\overline{fd}} \geq 2y_{bd}$$

$$x_{\overline{hi}} + x_{\overline{if}} + x_{\overline{fd}} \geq y_{bd}$$

The constraints on the other candidate paths are generated in the same manner. While constraints (3) ensure the coverage for individual sub-paths to cover multiple sub-paths, constraints (4) are required to ensure a continuous location. Constraints (4) can be read similarly to constraints (3): the sum of the selected RSs in A' must be greater than $K_{A'W'}$ if all OD pairs from W' are covered. Constraints (4) must be dealt with care given two important points: *i*) the total amount grows very quickly, and *ii*) it may not ever be necessary, given the network topology.

Fig. 3 illustrates its occurrences. Considering $R = 30$, $T = \{a, b, c, d\}$ and we only wish to cover the pairs (a, b) and (c, d) (i.e. $y_{ab} = y_{cd} = 1$ and $y_{ac} = y_{ad} = y_{bc} = y_{bd} = 0$). Following the reasoning in the previous paragraph, we have only two valid sub-paths: $\overline{aefb}$ and $\overline{cefd}$. Each one only requires one RS, and the only edge in common is $\overline{ef}$; naturally, the optimal solution that minimizes the budget spent is to locate one RS in $\overline{ef}$ satisfying all constraints (3).

However, there is no point in $\overline{ef}$ that installing one RS generates a spacing shorter than R between the travels for (a, b) and (c, d) (Section 4.3.2). In order to generate the appropriate constraint (4), given the graph G and a current cover $C = \{\overline{ef}\}$, the algorithm called Network Compression (A) has as output the needed sets ($W' = \{(a, b), (c, d)\}$ and $A' = \{\overline{ce}, \overline{ef}, \overline{fb}\}$) and the constant $K_{A'W'} = 2$. Then, we can now affirm that for ODs $W' = \{(a, b), (c, d)\}$ to be covered, edges $A' = \{\overline{ce}, \overline{ef}, \overline{fb}\}$ require at least $K_{A'W'} = 2$ RSs, leading to the corresponding constraint (4):

$$x_{\overline{ce}} + x_{\overline{ef}} + x_{\overline{fb}} \geq 2(y_{ab} + y_{cd} - 1)$$

To exemplify a case of point (*ii*), if edge $\overline{ef}$ is split in the middle in $\overline{ex}$ and $\overline{xf}$, the new solution that covers all sub-paths is one RS on $\overline{ex}$ and another on $\overline{xf}$, generating a feasible continuous location. Thus, no constraint (4) would need to be added to the formulation. We address points (*i*) and (*ii*) by considering only those continuous locations that do not violate any constraints specified in (3).

3.3. Model adaptations

Our model can be adapted to replicate previous models' results. The first and most common occurrence in other literature is the RS location on nodes, and the second is the split flow per OD, to change the trip coverage per OD to per path once we have multiple paths per OD pair. The third is the different initial or final AFV charge states. Most of the literature so far locates the RSs under the premise that an AFV leaves the origin with half of its total charge and must reach the destination with at least 50% charge. These are simple modifications as stated:

- **Location on Nodes**: Only additional variables are required for this adaptation, one binary variable for each node in the network, indicating if an RS is located in such a node or not. The sum of all new variables is added to z_1, since z_1 represents the total number of RSs located. Analogously, each Constraint (3) has the left side added one node variable to each node on sub-path s, excluding the origin and destination ones, and to locate on nodes of carnality equals 2 would be unnecessary;
- **Flow captured by a single covered path**: Some authors, when dealing with deviation paths, consider an OD coverage rule. Since the driver is aware of the refuel stations ahead, it is only necessary to cover one path to capture the OD path flow. Our model can be adapted to this premise by adding a variable for each path indicating whether it is covered, and a constraint for each OD pair that ensures the corresponding OD is covered if any path is covered. And another sub-path violation procedure needs to be written.
- **Split Flow**: The same changes applied at Kim and Kuby (2012) are applicable with the same logic, moving from covering a flow (trips) per OD to a flow per path, this requires a y variable per path and that all paths are enumerated;
- **Different initial or final charge states**: Many papers in literature have 50% of charge as an initial and final condition; this is the same approach as ours with half of the range. Initial or final charge conditions would only change the K_s calculation rules during the sub-problem (4.3). Using the 50% initial and final condition as an example, all sub-paths containing the origin or destination would have a K_s equal to $\left\lfloor \frac{|s|+R/2-\epsilon}{R} \right\rfloor$.
- **Known routes**: If the decision maker already has fixed routes between the ODs, one or many, there is only a need to change the sub-problem (4.3). With known paths, separating any violated sub-paths with a simple quadratic algorithm per path is possible.
- **Non-installable segments**: It is expected for any real road network to have multiple road segments not available for RS installation. For example, any edge $\overline{ab}$ of length 1, if the segment between 0.3 and 0.5 is not available, $\overline{ab}$ must be split into 3 segments, $\overline{ax}$ with length 0.3, $\overline{xy}$ with length 0.2 and $\overline{yb}$ with length 0.5 and a corresponding constraint of the forbidden segment $\overline{xy}$ must be added to the model $x_{\overline{xy}} = 0$.

4. Solution methods

Our approach is a bi-objective problem with two conflicting objectives that must be optimized simultaneously. Mathematical models can be used to solve such problems, and several methods are available. Multi-objective problems have a Pareto front, a set of non-dominated solutions that represent the trade-offs among the objectives. Solutions on the Pareto front are considered optimal if no solution can be improved in one objective without degrading another (Deb and Deb, 2014).

A solution of the SPFRLM is a cover $C = \{x_a | a \in A\}$ and a set of positive integer values representing the number of RSs for each edge $e \in A$ and, consequently, a set of OD pairs that C claims to cover $W_C := \{(o, d) | y_{od} = 1\}$. Thus, $z_1(C)$ is the total number of RSs suggested by C, and $z_2(C)$ is the total number of trips covered by C.

One notable number that appears multiple times when solving this problem is $C_{\min} := z_1(F_1(\beta = 1.0))$, the minimum number of RSs to cover all trips, which is the first number to be computed when solving an instance, it is used as a reference for the hyper-volume, and has one important proprety for computing a true pareto front.

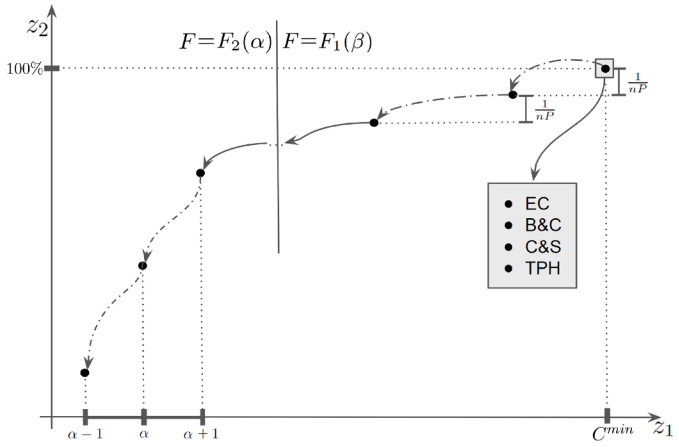

Fig. 4. Smoothest Descent Algorithm timeline.

Proposition 1. *A true Pareto front can not have more than $C_{\min}$ points.*

Proof. Let S be the set of Pareto-optimal solutions in the objective space (z_1, z_2), where z_1 minimizes the number of RSs and z_2 maximizes coverage. First, we observe that z_1 is strictly integer-valued ($x_a \in \mathbb{Z}^+$). For any specific integer value k of z_1, there can exist at most one non-dominated point in S. If two solutions exist with $z_1 = k$ and coverages z_2^a and z_2^b where $z_2^a > z_2^b$, the latter is dominated. If $z_2^a = z_2^b$, they map to the same point in the objective space. Second, by definition, $C_{\min}$ is the minimum number of RSs required to cover all trips (the maximum possible value of z_2). Any solution with $z_1 > C_{\min}$ cannot achieve higher coverage than the total number of trips Γ. Consequently, a solution with cost $C_{\min} + \delta$ (where $\delta \geq 1$) and coverage Γ is dominated by the solution $(C_{\min}, \Gamma)$. Therefore, the feasible domain for z_1 in the Pareto set is limited to the integer set $\{1, 2, \ldots, C_{\min}\}$. The cardinality of this set is $C_{\min}$. Thus, $|S| \leq C_{\min}$. □

Next, we propose a bi-objective algorithm (Section 4.1) to solve the FRLP. Furthermore, we propose four cover methods to solve each ϵ-constrained case in Section 4.2, in which some solution methods require a separation problem detailed in Section 4.3.

4.1. Smoothest descent algorithm

We propose and implement the Smoothest Descent Algorithm (SDA), which we can classify as a bi-objective heuristic ϵ-constraint algorithm for the FRLP, with a maximum number of Pareto solutions nP. If $nP = +\infty$, our proposed algorithm is an exact ϵ-constraint method. It is called SDA since it starts to solve the FRLP from the point with the smallest z_1 required to maximize z_2 (highest point if you visualize the solutions plane $z_1 \times z_2$ with z_2 as the vertical axis - Fig. 4) and only descents towards the origin (0,0) minimizing the z_2 variation with a z_1 step inversely proportional to nP.

An ϵ-constraint method places a constraint (7) or (8) on one of the objectives (1) or (2), respectively, and the other objective is optimized subject to that constraint. The constraint is then relaxed, allowing for a more flexible trade-off between the objectives. The SDA exploits the FRLP characteristics to heuristically solve fewer ϵ-constrained problems more efficiently than the exact method. For example, for each test problem, δ, and R, there is a corresponding $C^{\min}$. Fig. 4 illustrates an SDA timeline from right to left. It calculates first $C^{\min}$. Then it computes the remaining Pareto points with a step size of $1/nP$. If the number of remaining points exceeds the number of non-dominated points, the objective, ϵ-constraint, and steps are swapped, thereby saving CPU time and maximizing the hyper-volume. Fig. 4 also highlights that each solution can be calculated with any coverage method. We test four coverage methods: EC, B&C, C&S, and TPH, further detailed in Section 4.2.

Mainly, we work with two mathematical formulations: F_1 :=(1),(7),(3)–(6), a partial set cover formulation, and F_2 :=(2),(8),(3)–(6), a max cover formulation. Hence, formulation F_1 has β and is independent of α, and reciprocally, formulation F_2 has α as a parameter and is independent of β.

Algorithm 1 details the SDA. First, it receives the input data defined at Section 3.1 and a maximum number of Pareto points nP. Then, it starts by calculating $D_\omega^{\min}$ $(\omega = (i, j) | \forall i, j \in N)$ from and to all nodes in the network using a standard shortest path algorithm (Line 1), initialize the constant *step* (Line 2), the scalar variables α, β (Line 3), the current formulation F and the *Pareto* set PS (Line 4).

Then, the main loop occurs between Lines 5 - 11, in each iteration of, a *Cover* algorithm is used to find solutions C for the current formulation F (Line 6), where multiple possible *Cover* algorithms are presented in Section 4.2. The amount of RSs of C is stored in α, and C is added to the set of Pareto solutions PS (Line 7). In line 8, the current formulation can change if the current formulation is F_1 and $\alpha \leq nP$. This fact means the number of solutions to be calculated is higher than the total number of selected RSs and is more effective to calculate $\alpha - 1$ solutions with F_2 as the current formulation. If the current formulation is F_1, nP and β are decremented (Line 9), otherwise α is decremented (Line 10). The main loop ends if α or β reaches zero (Line 11). At Line 12, the resulting *Pareto* set is returned, and the algorithm ends.

Algorithm 1: Smoothest Descent Algorithm.

Data: Maximum number of Pareto solutions nP.
Result: A set PS of Pareto solutions.
1 Calculate all $D_\omega^{\min}$;
2 $step := \frac{1}{nP}$;
3 $\alpha \leftarrow +\infty$; $\beta \leftarrow 1.0$;
4 $F \leftarrow F_1(\beta)$; $PS \leftarrow \emptyset$;
5 **repeat**
6 $\quad C \leftarrow Cover(F, \alpha, \beta)$;
7 $\quad \alpha \leftarrow z_1(C)$; $PS \leftarrow PS \bigcup \{C\}$;
8 $\quad$ **if** $F = F_1$ *and* $\alpha \leq nP$ **then** $F \leftarrow F_2(\alpha)$;
9 $\quad$ **if** $F = F_1$ **then** $nP \leftarrow nP - 1$; $\beta \leftarrow \beta - step$;
10 $\quad$ **if** $F = F_2$ **then** $\alpha \leftarrow \alpha - 1$;
11 **until** $\alpha = 0$ *or* $\beta = 0$;
12 **return** PS

Proposition 2. *The SDA generates a true Pareto front if $nP \geq C^{\min}$ and all Pareto solutions are optimal.*

Proof. Let $\mathcal{Z}_1$ be the domain of the objective function z_1 (number of RSs). Since z_1 represents a count of discrete facilities, $\mathcal{Z}_1 \subset \mathbb{Z}^+$. The true Pareto front is defined by the set of non-dominated solutions $\{(z_1^*, z_2^*) \mid z_1^* \in [1, C^{\min}] \cap \mathbb{Z}\}$. The SDA is initialized by solving formulation $F_1(\beta = 1.0)$, yielding the solution with complete coverage and minimum necessary cost, such that the initial $z_1(C) = C^{\min}$. Given the condition $nP \geq C^{\min}$, the switching criteria $\alpha \leq nP$ (Line 8) is satisfied immediately after the first iteration. Consequently, the formulation switches to $F_2(\alpha)$, which maximizes coverage z_2 subject to a budget constraint $z_1 \leq \alpha$. In subsequent iterations, the algorithm decrements α by exactly 1 (Line 10) until $\alpha = 0$. Since the solver is assumed to return optimal solutions for each sub-problem $F_2(\alpha)$, the algorithm explicitly computes the optimal coverage z_2^* for every possible integer budget α in the feasible range $\{C^{\min}, C^{\min} - 1, \ldots, 1\}$. By exhaustively evaluating the optimal z_2 for every feasible discrete value of z_1, the resulting set PS necessarily contains all non-dominated points in the solution space. Thus, PS constitutes the true Pareto front. □

4.2. Cover algorithms

The SDA uses a decoupled coverage algorithm to generate each Pareto solution for a given formulation F and for each value of α or

β. This algorithm solves it and returns a cover C, which can be generated using either a heuristic or exact method. In our paper, all proposed algorithms are exact methods that utilize a mixed-integer programming (MIP) solver and a total time limit (TL_1). By using these algorithms, we can generate lower bounds and prove if the resulting cover is optimal. The MIP solver is a $Solver$ function that receives a formulation F ($F_1(\beta)$ or $F_2(\alpha)$), a set of constraints (3) and (4) D, a time limit TL, and returns a cover C. We provide a detailed explanation of our proposed cover algorithms in the following subsections.

4.2.1. Cut-and-Solve

The Cut-and-Solve (C&S) algorithm has two main steps: Solving and Cutting. Algorithm 2 formally describes our proposed C&S. It starts a current formulation with no constraints (3) or (4) (Line 1). In Solving, the MIP solver solves a static formulation with a smaller time limit TL_2 (Line 3), and Cutting solves the separation problem, which will be detailed in Section 4.3, for the best solution found during the Solving step (Line 3). If feasible ($D_2 = \emptyset$, i.e. C does not violate constraints (3) or (4)), C&S terminates (Line 2), otherwise the returned constraints are added to the current formulation (Line 5), and the Solving step is repeated. C&S has a disadvantage in finding feasible solutions for the more challenging instances as it terminates with none if TL_1 is reached (Line 7).

Algorithm 2: Cut-and-Solve.

Data: Formulation F and time limit TL_1.
Result: A cover C.
1 $D \leftarrow \emptyset$;
2 **repeat**
3 $\quad C \leftarrow Solver(F, D, \min(TL_2, TL_1 - Time))$;
4 $\quad D_2 \leftarrow SeparationProblem(C)$;
5 $\quad D \leftarrow D \cup D_2$;
6 **until** $D_2 = \emptyset$ *or* $Time \geq TL_1$;
7 **if** $Time \geq TL_1$ **then** $C \leftarrow \emptyset$;
8 **return** C

4.2.2. Branch-and-Cut

Branch-and-Cut (B&C) combines two methods for solving MIP problems: branching and cutting (Nemhauser and Wolsey, 1988). The branching step involves dividing the problem into smaller sub-problems by making decisions about the integer variables. In contrast, the cutting step involves adding valid inequalities to the problem that further restrict the feasible region. In our case, the MIP solver handles the branching, while the separation problem solves the cutting step (Section 4.3). Hence, our branch-and-cut implementation gradually adds the violated constraints of each new cover found. If the new cover does not separate any constraint, the cover becomes the current one. Otherwise, the new constraints are added to the model and the new cover is discarded.

The B&C implementation runs the separation procedure for valid solutions during the MIP Solver execution. Algorithm 3 starts the MIP solver with a set of constraints D (Line 1), and Lines 2-7) occurs simultaneously to Line 1 execution. Lines 3-7 are executed each time the MIP solver finds a cover C_1 with the best objective than the current best cover C, C_1 is verified by the $SeparationProblem$ (Line 3), if C_1 is valid ($D_2 = \emptyset$), C_1 becomes the new best cover found C (Line 5), otherwise the set of constraints is updated with the ones violated by C_1 (Line 7) and C_1 is discarded.

4.2.3. Enumerated coverage (EC)

This method is the simplest approach, usually employed by most FRLP studies (Kim and Kuby, 2012). Algorithm 4 presents the Enumerated Coverage pseudo-code, which enumerates all valid constraints (Line 1, Algorithm 5) and then solves the resulting formulation with a MIP solver (Line 3). For the SPFRLM, the procedure first runs Algorithm 5 to enumerate all candidate paths, then applies all constraints

Algorithm 3: Branch-and-Cut.

Data: Formulation F, a set of constraints D and time limit TL_1.
Result: A cover C.
1 $C \leftarrow Solver(F, D, TL_1)$);
2 **for** *Each* C_1 *better than the best* C *found* **do**
3 $\quad D_2 \leftarrow SeparationProblem(C_1)$;
4 $\quad$ **if** $D_2 = \emptyset$ **then**
5 $\quad\quad C \leftarrow C_1$;
6 $\quad$ **else**
7 $\quad\quad D \leftarrow D \cup D_2$;
8 **return** C

(3) and stores the resulting set in D. The MIP solver solves the resulting set of constraints D using F and the remaining CPU time.

Algorithm 4: Enumerated Coverage.

Data: Formulation F and time limit TL_1.
Result: A cover C.
1 $D \leftarrow Sub-pathEnumeration()$;
2 **if** $Time \leq TL_1$ **then**
3 $\quad C \leftarrow Branch-and-cut(F, D, TL_1 - Time)$
4 **else**
5 $\quad C \leftarrow \emptyset$
6 **return** C

Algorithm 5: Sub-path Enumeration.

Data: Global parameters and problem data
Result: Set with all constraints (3) D.
1 $D \leftarrow \emptyset$;
2 **for** $(o,d) \in W$ **do**
3 $\quad$ **for** $p \in Paths(o,d) | |p| \geq R$ *and* $|p| \leq D^{\max}_{(od)}$ **do**
4 $\quad\quad$ **for** $s \in Subpaths(p) | |s| \geq R$ **do**
5 $\quad\quad\quad D \leftarrow D \cup \{$Constraint (3) of $(s,(o,d))\}$
6 **return** D

Complexity Analysis. The computational complexity of the *Sub-path Enumeration* algorithm (Algorithm 5) is determined by the topology of the network and the density of the constraints. Let $|W|$ denote the number of OD pairs, and let Λ represent the maximum number of edges in any feasible candidate path (bounded by $|N|$).

Time Complexity: The algorithm iterates over all $w \in W$. The function $Paths(o,d)$ utilizes a Depth-First Search (DFS) to enumerate all simple paths satisfying the length constraint $D^{\max}_{(od)}$. In the worst case, the number of simple paths between two nodes is $O(|N|!)$. The DFS procedure requires $O(\eta \cdot \Lambda)$ time to generate these paths. Subsequently, the function $Subpaths(p)$ performs a quadratic forward analysis on each path p, generating $O(|p|^2)$ sub-paths. Since the worst-case scenario for a sub-path is to traverse all edges ($|p| = |E|$), the worst-case time complexity is dominated by the number of paths and is given by:

$$O\left(|W| \cdot |N|! \cdot |E|^2\right) \tag{9}$$

Space Complexity: The space complexity is dominated by the storage of the constraint set D. For every path p, up to $O(|E|^2)$ sub-path constraints are generated. In the worst case, assuming all generated sub-paths are unique and stored, the size of D grows linearly with the total number of enumerated sub-paths. Thus, the space complexity is:

$$O\left(|W| \cdot |N|! \cdot |E|^2\right) \tag{10}$$

4.2.4. Two-Phase hybrid (TPH)

We also propose a Two-Phase Hybrid (TPH) algorithm that combines the strategy of a Cut-and-Solve phase followed by a Branch-and-Cut phase. The C&S is very efficient in finding good dual bounds, and the B&C is more effective in finding better primal bounds when it starts the branching process from a set of constraints with a higher dual bound. Algorithm 6 details our TPH implementation. Lines 1 - 7 are the C&S phase, differs from Algorithm 2 at stopping criteria (Line 8) and the added Line 4 that stores the previous lower bound (LB_{prev}) found at Line 5. The current lower bound (LB_{curr}) and $Solver$ execution time T_{solver} are stored at Line 5. The added stopping criterion is if the lower bound does not change between iterations. If T_{solver} reaches the time limit TL_2, the same is compared against the stopping criteria if the lower bound does not improve between iterations. After the C&S phase ends, the TPH starts a B&C (Line 9) with the resulting set of constraints D and the remaining CPU time available if the C&S phase did not generate an optimal result ($D_2 \neq \emptyset$).

Algorithm 6: Two-Phase Hybrid.

Data: Formulation F and time limit TL_1.
Result: A cover C.

1 $D \leftarrow \emptyset$;
2 $LB_{curr} \leftarrow 0$;
3 **repeat**
4 $\quad LB_{prev} \leftarrow LB_{curr}$;
5 $\quad C, T_{solver}, LB_{curr} \leftarrow Solver(F, D, \min\{TL_2, TL_1 - Time)\}$;
6 $\quad D_2 \leftarrow SeparationProblem(C)$;
7 $\quad D \leftarrow D \cup D_2$;
8 **until** $D_2 = \emptyset$ *or* $Time \geq TL_1$ *or* $T_{solver} = TL_2$ *or* $LB_{prev} = LB_{curr}$;
9 **if** $D_2 \neq \emptyset$ **then** $C \leftarrow B\&C(F, D, TL_1 - Time)$;
10 **return** C

4.3. Separation problem

When a model has too many constraints to enumerate, a separation problem is used to determine whether a solution is feasible. It takes a solution as input, and either returns a set of violated constraints or validates the input solution as feasible (Dantzig and Wolfe, 1960).

Fig. 5 shows the chain of decisions of the separation problem. Given a cover C, the separation problem first checks for a violated sub-path (Section 4.3.1); if any is found, the paths are stored, and the violated constraints (3) are returned, otherwise a continuous location (Section 4.3.1) is verified; if negative, at least a constraint (4) is returned, otherwise C is a feasible solution and has a continuous location.

4.3.1. Sub-path violation

Given a cover C, for each OD pair $(o,d) \in W_C$ that C claims to cover ($W_C := \{(o,d)|y_{od} = 1\}$), a Formulation (11)-(24) is solved to check if there is a valid sub-path from a valid path from o until d that C does not cover. If such a sub-path s exists, it returns s; otherwise, it returns an empty set. The separation problem does not need to search for sub-paths in the original graph G. To optimize the search, it searches on sub-graph $G^{od} = \{N^{od}, A^{od}\} \subseteq G$, with set of nodes $N^{od} := \{i | i \in N,\ D^{\min}_{oi} + D^{\min}_{id} \leq D^{\max}_{od}\}$ and set of edges $A^{od} := \{\overline{ij} | \overline{ij} \in A,\ i,j \in N^{od}\}$. For each node i, we have $A^{od}_i := \{\overline{kl} | \overline{kl} \in A^{od},\ k = i \vee l = i\}$, the set of edges that contains i. And $\Psi(T)$ is the number of elements of set T.

Let K^{od}_i be constants where $K^{od}_i = 1$ if $i = o$ or $i = d$, and 0 otherwise. Additionally, $|a|$ represents the length of edge a, and C_a indicates the number of RS located on edge a. For binary variables, $\chi^p_a = 1$ indicates that edge a is part of a candidate path $p = \{a|\chi^p_a = 1\}$ between o and d. Likewise, χ^s_a indicates if edge a is part of a valid sub-path between o and d. Variables γ^p_i and γ^s_i equal 1 if i is an intermediary node on path p or sub-path s, respectively. Finally, $\omega_i = 1$ indicates that node i is the origin or destination, respectively, of the selected sub-path, 0 otherwise, as the selected sub-paths do not contain a fixed origin or destination.

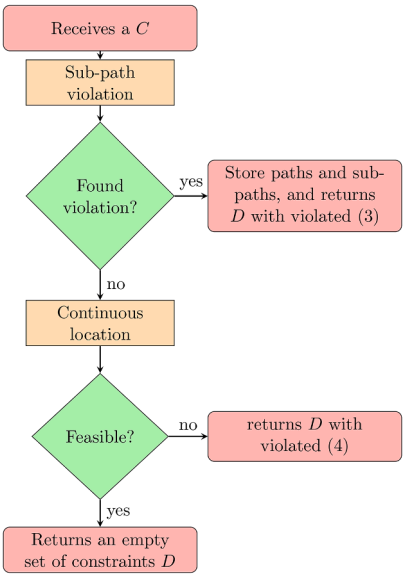

Fig. 5. Separation problem flowchart.

$$\sum_{a \in A^{od}_i} \chi^p_a = 2\gamma^p_i \qquad \forall i \in N^{od} \setminus \{o,d\} \tag{11}$$

$$\sum_{a \in A^{od}_i} \chi^p_a = 1 \qquad \forall i \in \{o,d\} \tag{12}$$

$$\sum_{\overline{ij} \in A, i,j \in T, i \neq j} \chi^p_{\overline{ij}} \leq \Psi(T) - 1 \qquad \forall T \subseteq N^{od}, \Psi(T) \geq 3 \tag{13}$$

$$\sum_{a \in A^{od}} |a| \chi^p_a \leq D^{max}_{od} \tag{14}$$

$$\chi^p_a \leq \chi^s_a \qquad \forall a \in A^{od} \tag{15}$$

$$\sum_{a \in A^{od}_i} \chi^s_a = 2\gamma^s_i + \omega_i \qquad \forall i \in N^{od} \tag{16}$$

$$\gamma^s_i + \omega_i \leq 1 \qquad \forall i \in N^{od} \tag{17}$$

$$\sum_{i \in N^{od}} \omega_i = 2 \tag{18}$$

$$\sum_{a \in A^{od}} C_a \chi^s_a - k \leq -1 \tag{19}$$

$$\sum_{a \in A^{od}} |a| \chi^s_a - Rk \geq 0 \tag{20}$$

$$\sum_{a \in A^{od}} |a| \chi^s_a - Rk \leq R \tag{21}$$

$$\chi^p_a,\ \chi^s_a \in \{0,1\} \qquad \forall a \in A^{od} \tag{22}$$

$$\gamma^p_i,\ \gamma^s_i,\ \omega_i \in \{0,1\} \qquad \forall i \in N^{od} \tag{23}$$

$$k \in \mathbb{Z}^+ \tag{24}$$

Formulation (11)-(24) can be observed in three parts, the first one (11)-(14) searches for a path $p = \{(a|\chi^p_a = 1\}$ between o and d. Constraints (11) manage the cardinality of each node i, which can be an intermediary, for the path. Constraints (12) force o and d cardinality as 1. Constraints (13) ensure no sub-tours. And Constraint (14) fixes the maximum path size as $D^{\max}_{od}$.

The second part (15)-(18) searches for a sub-path $s = \{a|\chi^s_a = 1\}$ between $i|\omega_i = 1$ and $j|\omega_j = 1$ (Constraints (16)-(18)). Constraints (15) ensure any edge selected for the sub-path is selected on the path. Constraints (16) manage the cardinality of each node i for the sub-path.

Constraints (17) ensure no intermediary node is an origin or destination. Constraint (18) ensures that the sub-path has two origin and destination nodes.

The third and last part is (19)-(21), Constraint (19) ensures that the selected sub-path s violates Constraint (3) with C. Constraints (20) and (21) ensure that $k = \left\lfloor \frac{|s|-\epsilon}{R} \right\rfloor$. Finally, Constraints (22)-(24) define the domains of the variables. If feasible, s separates a constraint (3) and p is stored for future continuous locations.

Complexity. For a given O-D pair, we can characterize the complexity of the separation problem by decomposing it based on the feasible number of refuelling intervals k, such that $D_{od}^{\min} \leq kR \leq D_{od}^{\max}$. For each fixed k, a sub-problem is a K Single Resource Constrained Shortest Path (SRCSP) problem. In this formulation, the "resource" is defined by the edges equipped with RSs, which are limited to at most $k-1$ traversals. Additionally, the path must satisfy duration (or length) constraints, specifically being strictly greater than $k \times R$ and less than or equal to $D_{od}^{\max}$, which is solved by finding the k shortest ones. As established in the literature, the SRCSP is NP-complete (Festa, 2015), and the problem of finding its k shortest solutions was studied in Mehlhorn and Ziegelmann (2000) and Liu and Ramakrishnan (2001).

Computational optimizations. Given that Formulation (11) - (24) is a difficult problem to solve, and we did some computational optimizations to make the sup-problem solution more efficient:

- As previously described by using sub-graph G^{od};
- For each $od \in W$, Formulation (11)-(24) is mostly static, except for Constraint (19), therefore each model per od is stored in memory after loading the instance and Constraint (19) is updated for each cover C;
- Whereas each separation model is stored in memory and reused, all enumerated sub-tours $st \subseteq N^{od}$ stay in the model for the following covers C.

4.3.2. Continuous location

In analyzing a cover C that satisfies all sub-path constraints, ensuring a continuous location is the final validation step for C to be considered a feasible solution. A continuous location is deemed feasible when the RSs specified by C are positioned such that, for any candidate path p covered by C, the distance between successive RSs along p does not exceed the threshold R. The formulation (11)-(24) is used to verify the existence of valid continuous locations. A positive outcome confirms the validity of cover C, and a set of located RSs, denoted as $L = l_{a_k}$, is identified. Conversely, if no valid continuous location is found, Algorithm 7 is employed to separate a constraint, as defined in (4).

For the continuous location of cover C, let continuous variable $l_{a_k} \in [0,1]$ with $k \in \{1, \ldots, C_{\overline{a}}\}$ for each corresponding RS located on edge $\overline{a}$, that indicates the relative position of RS a_k on $\overline{a}$. Let $LP_{p,C}$ be the set of consecutive pairs of RS nodes along path p with cover C. For each path p and RS node i, π_{ip} is the relative position of i along path p, and variable r_w indicates the amount of relaxation in OD pair w for a continuous location to exist. Each edge $\overline{a}$ and OD pair w have consistent starting and final points that determine each π_{ip} calculation, thus each π_{ip} can take three different forms: (i) if i is o or d (a pre-existent RS location) or; (ii) if i is on an edge parametrized in the same way as p or; (iii) if i is on an edge parametrized in the opposite way as p. In case (i), π_{ip} is a constant, and cases (ii) and (iii) will be exemplified below.

The objective function (25) is the sum of all OD relaxations; $z_3 = 0$ indicates a feasible continuous location, otherwise unfeasible. Our only set of constraints (26) ensures a distance $\leq R$ for each pair of RSs along all covered paths, given no relaxation. Since not all paths may be enumerated. For each feasible location L, it is necessary to check for violated constraints (26) with the Separation problem for continuous location B.

$$\text{Minimize } z_3 = \sum_{w \in W_C} r_w \tag{25}$$

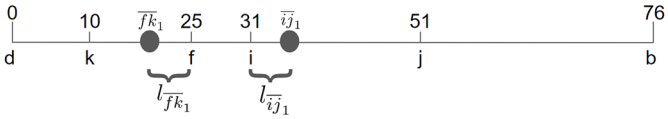

Fig. 6. Continuous location - example of variables.

Subject to:

$$\pi_{jp} - \pi_{ip} + r_w \leq R \qquad \forall (i,j) \in LP_{p,C}, p \in P_w, w \in W_C \tag{26}$$

$$r_w \in \mathbb{R}^+ \qquad w \in W_C \tag{27}$$

$$l_{a_k} \in [0,1] \qquad \forall k \in \{1, \ldots, C_a\}, \forall a \in A. \tag{28}$$

Fig. 6 takes the path example of Fig. 2, with a suggested location of an RS on edge fk and another on ij on path $p = \overline{dkfijb}$, $LP_{p,C}$ is then $\{(d, \overline{fk}_1), (\overline{fk}_1, \overline{ij}_1), (\overline{ij}_1, b)\}$. Let the location of edge ij follow the same direction of path p and edge fk the opposite one. This implies $l_{\overline{fk}_1}$ being the relative position of the suggested RS from f to k and the same for $l_{\overline{ij}_1}$ from i to j, therefore the $\pi_{\overline{fk}_1,p} = |\overline{dk}| + |\overline{fk}| \times (1 - l_{\overline{fk}_1})$ and $\pi_{\overline{ij}_1,p} = |\overline{dkfi}| + |\overline{ij}| \times l_{\overline{ij}_1}$.

5. Experimental results

All the algorithms are coded in C and compiled with GCC. The MIP solver used is Gurobi 11.02 (Gurobi Optimization, 2024). The computer used in all experiments is an AMD Ryzen 7950X 16-core/32-thread processor, 96GB of DDR5 RAM, and ran Ubuntu 24.04 x64.

We evaluate the performance of our models and algorithms on a comprehensive set of 26 real-world instances derived from a road transportation network. This network is extracted from a georeferenced database publicly available via the Brazilian National Department of Transport Infrastructure, dated 2021. Each instance corresponds to a state, with the origins and destinations represented by cities within that state. We deliberately select these instances because they reflect plausible scenarios encountered by government policymakers seeking to implement carbon-neutral initiatives that facilitate road travel using AFVs. Additionally, these instances are relevant to private decision-makers seeking to optimize refuelling strategies for road trips. The instances are named based on difficulty after most of our testing. Table 2 presents the number of nodes, edges, OD nodes, and OD pairs with the number of trips greater than zero. These instances are available at a *Github* repository[1].

Our hyper-volume indicator uses the point $(z_1^0, z_2^0) = (C_{\min}, 0)$ as the reference. For any given cover C, the area dominated by C is defined as the rectangle bounded by the vertices $(z_1(C), z_2(C))$ and (z_1^0, z_2^0). Consequently, for any set of solutions $PS = \{C_1, C_2, \ldots, C_n\}$ returned by the SDA, the hyper-volume $HV(PS)$ corresponds to the area of the polygon formed by the union of the dominated rectangular regions defined by every $C \in PS$. As a visual aid, Fig. 7 later in this section shows the representation of each polygon for a particular instance.

For all test, TL_1 and TL_2 are fixated at 14400 and 2000 seconds, respectively, $R = 100km$, $\delta \in \{1.01, 1.05, 1.10, 1.15\}$ and $nP \in \{+\infty, 100, 50, 25, 10\}$. The impact of multiple R values was studied by most FRLM literature, whereas there are fewer ones with deviation paths (Kim and Kuby, 2012; Yildiz et al., 2016), which implies that the parameter δ and its values were based on them. The nP tested values are based on initial internal testing, and we found them to be representative of the expected SDA behaviour.

5.1. Comparison of cover algorithms

As presented in Section 4.2, for each ϵ-constrained case, there is a formulation F paired with a value of α or β and all other parameters to

[1] https://github.com/brunosalezze/frlp-instances

Table 2
Instance data.

Instance	#Nodes	#Edges	#ODs	#OD Pairs	Instance	#Nodes	#Edges	#ODs	#OD Pairs
p01	32	32	14	91	p14	357	449	178	6775
p02	145	145	43	194	p15	273	313	178	9604
p03	73	76	15	105	p16	429	548	217	9640
p04	48	50	22	231	p17	432	604	182	9162
p05	132	139	51	1134	p18	471	594	277	22,607
p06	160	211	74	1564	p19	332	373	133	6910
p07	167	207	99	3763	p20	1015	1243	141	5954
p08	380	500	136	2183	p21	838	1076	404	19,445
p09	614	761	92	1838	p22	776	957	406	45,163
p10	303	410	78	2051	p23	888	1146	389	36,220
p11	366	452	217	9438	p24	1108	1516	244	17,049
p12	330	420	165	5299	p25	1749	2060	805	84,233
p13	435	576	78	1641	p26	1296	1669	612	91,295

Table 3
Results for formulation F_1, $\delta = 1.15$ and $\beta = 0.9$.

	TPH			C&S			B&C			EC		
Instance	UB	LB	Time (s)	UB	LB	Time (s)	UB	LB	Time (s)	UB	LB	Time (s)
p01	**11**	**11**	0.19	**11**	**11**	0.31	**11**	**11**	3.30	**11**	**11**	1.69
p02	**72**	**72**	0.63	**72**	**72**	0.30	**72**	**72**	6.51	**72**	**72**	8.75
p03	**11**	**11**	0.34	**11**	**11**	0.22	**11**	**11**	8.92	**11**	**11**	10.82
p04	**16**	**16**	14.29	**16**	**16**	10.38	**16**	**16**	1.18	**16**	**16**	11.26
p05	**16**	**16**	19.52	**16**	**16**	12.26	**16**	**16**	415.78	**16**	**16**	458.44
p06	**7**	**7**	13.969	**7**	**7**	17.96	**7**	**7**	456.64	**7**	**7**	745.17
p07	**14**	**14**	11013.73	**14**	**14**	8236.92	**14**	**14**	13354.92	**14**	**14**	13743.56
p08	**62**	**62**	2061.80	**62**	**62**	2624.35	**62**	**62**	6755.13	**62**	**62**	7564.68
p09	**31**	**31**	433.133	**31**	**31**	1254.14	33	29	14400.00	35	26	14400.00
p10	**25**	**25**	829.30	**25**	**25**	12846.19	26	25	14400.00	-	-	14400.00
p11	**38**	**38**	6450.97	-	38	14400.00	38	34	14400.00	50	32	14400.00
p12	**25**	**25**	11006.55	-	25	14400.00	25	21	14400.00	41	25	14400.00
p13	**47**	**47**	1820.02	**47**	**47**	11385.09	52	45	14400.00	61	45	14400.00
p14	29	**28**	14400.00	-	28	14400.00	36	23	14400.00	-	-	14400.00
p15	**70**	**70**	13006.37	-	70	14400.00	75	61	14400.00	78	60	14400.00
p16	**88**	**88**	13016.71	-	88	14400.00	92	85	14400.00	92	74	14400.00
p17	65	58	14400.00	-	60	14400.00	70	55	14400.00	70	45	14400.00
p18	68	64	14400.00	-	66	14400.00	73	55	14400.00	-	-	14400.00
p19	101	95	14400.00	-	95	14400.00	110	85	14400.00	-	-	14400.00

be solved. We conduct experiments to compare the proposed coverage methods: Enumerated coverage (EC), Branch and Cut (B&C), Cut and Solve (C&S), and Two-phase hybrid (TPH). Table 3 shows the best objective, in this case, the upper bound (UB) found by the SDA with each method, the dual bound, in this case, the lower bound (LB), and the total CPU time (Time) in seconds. We test these methods on 19 instances with formulation F_1, $\delta = 1.15$, and $\beta = 0.9$. These results indicate how well the proposed cover algorithms performed in other tests we conducted with different values of δ, β, or with formulation F_2 and multiple values of α. Tests that are optimally proven are highlighted in bold.

To statistically compare the performance of the algorithms presented in Table 3, we split the tested instances in two main groups, the ones with the same lower and upper bounds comparing time and the renaming for solution quality. For the computational efficiency analysis (instances $p01$–$p08$), we define test the null hypotesis for equality, upon rejecting it, using a Friedman test ($\alpha = 0.05$), we apply a one-tailed Nemenyi post-hoc test to evaluate the hypothesis of superiority in pairs. The results confirm that C&S is significantly superior in speed for small-scale instances. For the solution quality analysis (instances $p09$–$p19$), we focus on the Upper Bound (UB) values. After rejecting the null hypothesis of equal means, we test the superiority hypothesis in pairs. A Wilcoxon signed-rank test confirms the superiority of TPH ($p < .01$), as it consistently finds better upper bounds and is the only method to optimally solve several instances, such as $p11$ and $p12$, within the time limit. Furthermore, TPH demonstrates superior robustness compared to C&S and EC, which frequently fail to return any feasible solution for the hardest instances.

5.2. Bi-objective results and analysis

In this section, we analyze the SDA results, beginning with the tests in which all calculated Pareto points were optimal, as shown in Tables 4 and 5. Fig. 7 presents multiple nP configuration results and their effects. Figs. 8 and 9 visually analyze the expected difficulty for targeted coverage or RSs. With the results of Section 5.1, all SDA tests in this section are obtained using the TPH as the coverage method.

In Tables 4–7, we only show the results for the instances in which SDA was able to obtain the optimal Pareto fronts with multiple δ and nP values. Each table has a fixed δ, and we compare the calculated number of Pareto points, the relative hyper-volume (HV), and the CPU time to the exact Pareto value for each nP configuration. We can observe Proposition 1 implications since all results are optimal, $nP = +\infty$ implies $\#Points = C^{\min}$, and for any $nP \geq C^{\min}$, the execution time and hyper-volume are the same as the optimal one.

Out of the 26 test instances, the SDA calculates the Pareto optimal for 19 instances with $\delta = 1.01$ (Table 4), 16 instances with $\delta = 1.05$ (Table 5), 14 instances with $\delta = 1.10$ (Table 6), and 11 instances with $\delta = 1.15$ (Table 7). Other authors also show that the difficulty overgrows with higher values of δ (Lin et al., 2020; Zhang et al., 2019).

However, we observe that the SDA is highly efficient, achieving most of the hyper-volume with a fraction of the CPU time required for a Pareto-optimal ($nP = +\infty$). The large-scale instance results show that, for $nP = 25$, the SDA achieves $> 97\%$ of the optimal hyper-volume and, in most cases, spends $< 10\%$ of the CPU time. The $nP = 10$ results show

Table 4
Instances with optimal results for $\delta = 1.01$.

nP	$+\infty$		100			50			25			10		
	#Points	Time(s)	#Points	HV(%)	Time(%)	#Points	HV(%)	Time(%)	#Points	HV(%)	Time(%)	#Points	HV(%)	Time(%)
p01	11	0.01	11	100.00	100.09	11	100.00	99.18	11	100.00	99.54	7	93.44	64.95
p02	151	16.16	75	99.68	82.29	39	99.32	60.25	21	98.69	2.04	10	93.70	1.53
p03	13	0.04	13	100.00	99.60	13	100.00	99.03	13	100.00	99.73	7	98.11	35.55
p04	25	0.34	25	100.00	100.61	25	100.00	99.37	25	100.00	100.09	10	95.59	33.86
p05	27	108.08	27	100.00	100.46	27	100.00	100.28	11	96.93	96.00	8	92.61	1.59
p06	6	0.01	6	100.00	99.60	6	100.00	100.12	6	100.00	100.85	6	100.00	100.48
p07	16	2.74	16	100.00	99.28	16	100.00	100.07	16	100.00	100.31	5	92.32	35.16
p08	112	144.99	75	99.46	12.06	31	98.86	6.20	18	97.36	3.44	9	92.37	0.62
p09	26	2.58	26	100.00	99.32	26	100.00	100.44	18	96.01	85.37	6	92.68	18.46
p10	33	17.52	33	100.00	100.96	33	100.00	99.71	12	97.68	45.80	9	93.61	19.77
p11	42	1036.75	42	100.00	100.88	42	100.00	99.99	14	97.70	92.86	9	92.74	58.36
p12	33	20.65	33	100.00	100.34	33	100.00	99.87	13	97.01	82.11	9	92.50	37.11
p13	64	2805.50	64	100.00	99.34	26	98.96	8.50	17	97.78	6.55	9	93.48	0.15
p14	37	130.6	37	100.00	99.95	37	100.00	100.26	21	98.41	57.13	10	91.78	15.85
p15	79	6953.9	79	100.00	100.09	47	99.27	94.14	25	97.59	18.92	10	92.45	6.75
p16	89	26150.6	89	100.00	99.98	48	99.25	14.66	25	97.58	3.39	10	92.44	0.86
p17	69	23802.7	69	100.00	100.19	45	99.38	66.52	25	97.57	8.45	10	92.35	1.75
p18	64	7992.2	64	100.00	100.07	38	99.22	73.69	24	97.69	47.5	10	92.93	14.38
p19	140	35778.2	92	99.75	57.06	49	98.62	45.83	24	96.92	9.03	9	92.12	4.35
Averages				99.98	96.89		99.65	80.56		98.65	56.55		94.18	30.96

Table 5
Instances with optimal results for $\delta = 1.05$.

nP	$+\infty$		100			50			25			10		
	#Points	Time(s)	#Points	HV(%)	Time(%)	#Points	HV(%)	Time(%)	#Points	HV(%)	Time(%)	#Points	HV(%)	Time(%)
p01	11	0.7	11	100.00	100.05	11	100.00	100.67	11	100.00	100.21	6	92.31	64.45
p02	151	15	69	99.69	74.96	38	99.37	25.85	21	98.69	3.16	9	93.52	1.76
p03	13	0.6	13	100.00	99.91	13	100.00	100.25	13	100.00	100.87	7	98.57	54.50
p04	25	2.6	25	100.00	100.25	25	100.00	100.89	25	100.00	99.40	9	95.29	22.18
p05	27	26.2	27	100.00	100.51	27	100.00	99.20	12	97.01	93.50	7	92.74	3.67
p06	6	0.8	6	100.00	100.93	6	100.00	99.48	6	100.00	100.00	6	100.00	100.26
p07	16	71.7	16	100.00	100.39	16	100.00	99.20	16	100.00	100.32	5	91.04	63.01
p08	112	623.6	46	99.36	8.49	31	98.71	4.32	18	97.10	0.75	8	92.12	0.32
p09	26	203.2	26	100.00	100.02	26	100.00	100.78	9	96.83	34.16	6	93.86	19.73
p10	33	697	33	98.73	99.82	33	98.73	99.81	15	97.53	34.43	8	93.22	15.19
p11	42	1441.9	42	100.00	99.67	42	100.00	99.37	16	97.50	38.72	8	92.33	17.15
p12	33	416.8	33	100.00	99.81	33	100.00	100.63	14	96.55	31.76	7	91.37	15.34
p13	81	2416.4	81	100.00	100.21	26	98.73	14.50	17	97.33	6.93	8	92.50	0.29
p14	46	2456.5	46	100.00	100.47	46	100.00	100.57	22	98.07	25.12	9	92.75	8.26
p15	89	27005.8	89	100.00	100.15	50	99.33	61.04	24	98.13	12.74	10	92.87	3.68
p16	102	845231.7	93	99.55	99.95	46	99.11	9.57	22	96.41	5.42	10	93.38	1.78
Averages				99.97	95.19		99.75	80.07		98.71	50.72		94.28	29.32

that the SDA achieves > /92% of the optimal hyper-volume and, in most cases, spends < 5% of the CPU time.

In most cases, $nP = 100$ did not significantly improve the CPU times since only a few cases have $C^{\min} < 100$, so most SDA runs are exact however for $nP = 50$ we can already observe significant CPU time savings like for instances with higher $C^{\min}$ values like p08, p13, p16, and p19 while retaining at least 98.5% of the optimal hyper-volume.

Previous FRLP literature, such as Kim and Kuby (2012) and Yildiz et al. (2016), show that the difficulty (the CPU time required to solve it optimally) grows very quickly as δ increases. We also observe this behaviour for the optimal Pareto front in all tested instances, primarily from $p07$ onward, which is expected to be higher than the cost of solving a fixed number of points.

Table 8 shows the results for the hardest instances in which the SDA was not able to obtain an optimal Pareto, Column #Optimal counts the number of optimal solutions out of 10, Max Gap is the maximum percentile optimally gap, Time is the CPU time in seconds, and #Paths is the maximum number of calculated candidate paths per solution. These results show that the difficulty of solving it optimally can increase rapidly with the largest instances, since this is the easiest case possible; even then, four of the seven instances could not solve 10 points optimally. In all instances, at least half of the points were optimal.

Fig. 7 presents the Pareto fronts for instance $p19$ with $\delta = 1.01$. The figure visualizes the hyper-volume of each tested nP configuration as the polygonal area formed by the union of the rectangular regions defined by the solution points and the reference $(C_{\min}, 0)$. Due to the heuristic nature of the SDA when $nP < C^{\min}$, the generated curves for lower nP values (e.g., 10 or 25) represent approximations of the optimal frontier. Notably, the solution set for $nP = 100$ is virtually indistinguishable from the exact Pareto front ($nP = +\infty$), while the $nP = 25$ configuration closely follows the optimal trajectory, capturing approximately 97% of the optimal hyper-volume.

Figs. 8 and 9 present the relative difficulty to optimally solving each instance related to a target coverage (Fig. 8) or the target normalized RSs (Fig. 9). The relative difficulty is: for each optimal Pareto set resulting from an SDA execution with $nP = +\infty$, the relative difficulty for each Pareto point is the normalized solution time for the Pareto set. The target RSs (%) values in Fig. 9 are related to $C^{\min}$ and the target coverage (%) values in Fig. 9 are related to γ.

Observing Fig. 8, we can note that the difficulty of solving it is higher for target coverage in the interval $[80, 100]$%, with a peak

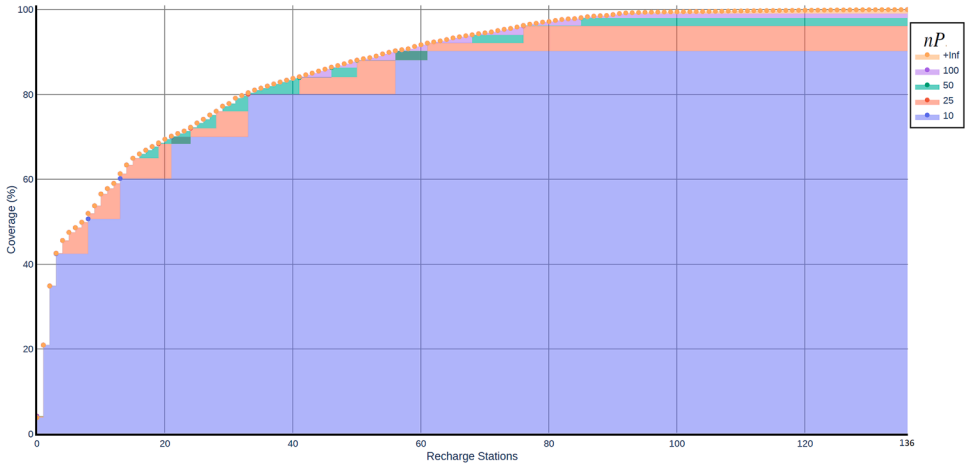

Fig. 7. Algorithm 1 results for p19 with $\delta = 1.01$ and multiple nP configurations and its hyper-volumes.

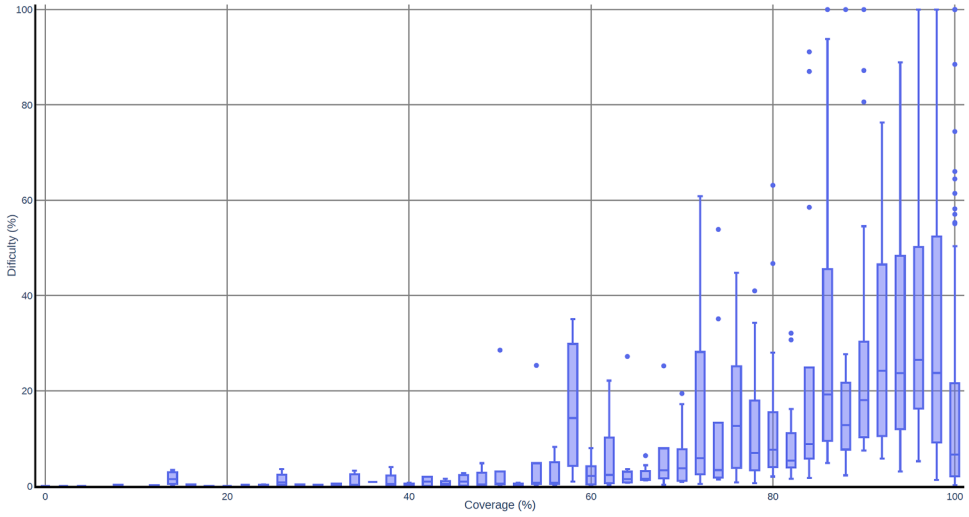

Fig. 8. Relative difficulty for 50 coverage intervals.

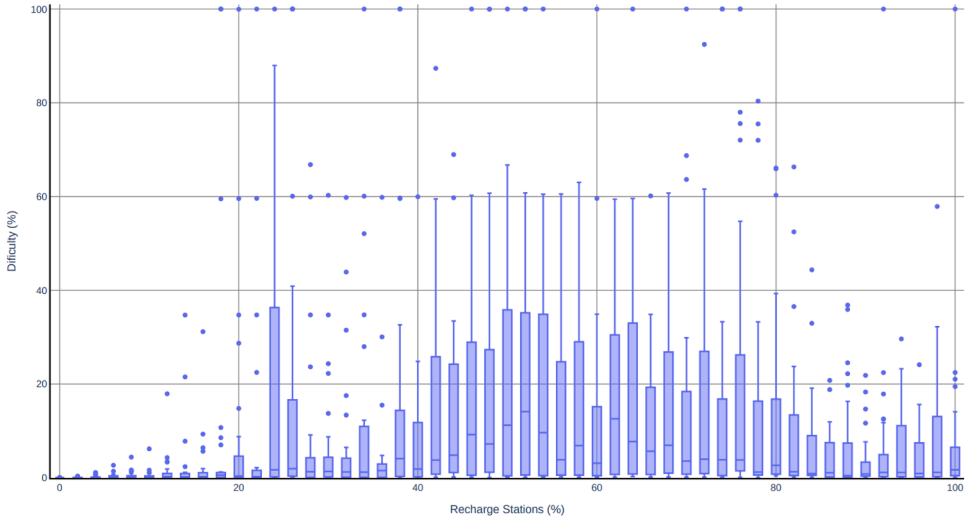

Fig. 9. Relative difficulty for 50 normalized targeted RSs intervals.

Table 6
Instances with optimal results for $\delta = 1.10$.

nP	$+\infty$		100			50			25			10		
	#Points	Time(s)	#Points	HV(%)	Time(%)	#Points	HV(%)	Time(%)	#Points	HV(%)	Time(%)	#Points	HV(%)	Time(%)
p01	13	1.2	13	100.00	100.42	13	100.00	99.49	13	100.00	100.65	6	91.37	33.08
p02	152	475.3	69	99.84	67.82	38	99.55	41.75	21	98.99	3.95	10	93.65	2.92
p03	14	6.8	14	100.00	99.46	14	100.00	99.71	14	100.00	100.64	9	98.85	74.33
p04	26	23.1	26	100.00	100.28	26	100.00	100.15	25	98.65	100.06	10	95.05	20.79
p05	29	415.8	29	100.00	99.24	29	100.00	100.59	12	97.26	47.11	7	93.33	17.62
p06	10	28.1	10	100.00	99.35	10	100.00	99.28	10	100.00	99.94	9	95.25	99.37
p07	21	2622.9	21	100.00	99.36	21	100.00	100.02	21	100.00	99.24	6	91.95	25.78
p08	135	55816.5	75	99.44	40.76	41	99.30	10.76	20	98.84	7.09	10	96.00	5.45
p09	36	20473.6	36	100.00	100.99	36	100.00	100.44	12	97.82	14.90	7	92.91	5.75
p10	42	63185.2	42	100.00	100.71	42	100.00	99.78	17	96.69	32.36	8	91.04	6.34
p11	51	187614.1	51	100.00	99.17	27	98.94	58.11	19	97.71	36.40	8	92.37	13.49
p12	44	56314.2	44	100.00	99.98	44	100.00	99.78	16	97.15	23.81	8	91.75	5.16
p13	56	948901.5	56	100.00	100.00	27	99.14	24.16	17	97.80	9.57	8	93.03	1.22
p14	63	145982.1	50	99.45	99.05	34	99.12	17.30	24	96.14	4.37	9	92.10	2.51
Averages				99.97	93.31		99.74	76.28		98.39	47.71		93.94	28.55

Table 7
Instances with optimal results for $\delta = 1.15$.

nP	$+\infty$		100			50			25			10		
	#Points	Time(s)	#Points	HV(%)	Time(%)	#Points	HV(%)	Time(%)	#Points	HV(%)	Time(%)	#Points	HV(%)	Time(%)
p01	13	0.5	13	100.00	98.07	13	100.00	100.00	13	100.00	99.42	10	98.06	127.94
p02	152	12.6	83	99.83	66.15	49	99.04	57.62	25	98.41	29.3	10	92.44	10.74
p03	14	2.1	14	100.00	99.86	14	100.00	100.72	14	100.00	102.25	10	95.49	164.88
p04	25	4.1	25	100.00	99.83	25	100.00	99.56	25	100.00	99.68	9	95.67	24.34
p05	30	144.7	30	100.00	100.06	30	100.00	100.42	18	98.56	31.88	10	94.04	10.00
p06	11	31.6	11	100.00	100.13	11	100.00	99.39	11	100.00	98.43	7	97.94	55.2
p07	26	1000.6	26	100.00	100.45	25	100.00	98.45	25	100.00	98.65	10	94.19	29.55
p08	172	97834.8	92	99.69	11.61	48	98.92	1.79	25	97.36	0.68	10	92.2	0.21
p09	54	15117.6	54	100.00	100.02	49	99.85	99.95	23	97.98	12.9	10	93.97	9.12
p10	61	53664.2	61	100.00	100.01	32	98.47	13.47	25	94.88	8.69	10	93.08	4.47
p11	58	275071.7	58	100.00	99.99	38	99.24	23.99	24	97.34	4.95	10	89.36	1.12
Averages				99.96	88.74		99.61	72.49		98.59	53.53		94.22	39.78

Table 8
Hardest instances results for $\delta = 1.01$ and $nP = 10$.

Instance	#Optimal	Max Gap (%)	Time(s)	#Paths
p20	10	0.00	1001.07	12,543
p21	10	0.00	9374.86	36,424
p22	9	1.96	29709.46	99,723
p23	8	6.00	45158.68	135,408
p24	10	0.00	4476.76	75,205
p25	5	22.86	77750.31	258,696
p26	6	28.13	63074.57	195,755

around 95%. On the other hand, it is the easiest for target coverage values below 50%. Also, the average difficulty in calculating $C^{\min}$ (100% coverage) is approximately 8%, which is not very high. Fig. 9 shows that the difficulty related to a relative target #RSs is not predictable in the target coverage results. Still, we observe that the problem is straightforward to solve up to 16% and not very difficult from $82 - 100\%$.

6. Discussions

The experimental results demonstrate that the proposed Smoothest Descent Algorithm (SDA), coupled with the Two-Phase Hybrid (TPH) method, successfully balances computational efficiency with solution quality. While the other tested methods, such as Branch-and-Cut, often find solutions on large-scale instances within the same time limits, the SDA consistently approximates the Pareto front with high accuracy. Specifically, we observed that with $nP = 25$, the system captures approximately 97% of the optimal hyper-volume while reducing computational time by over 50% on average and up to 96.5% on larger instances compared to the exact approach. This allows for strategic planning—where rapid scenario analysis is often more valuable than rigorous, exact ones, and the SDA provides a viable alternative.

The bi-objective nature of the results offers critical insights into the marginal cost of service coverage. Our "difficulty" analysis indicates that the financial and computational effort required to increase coverage from 80% to 100% is disproportionately high compared to early gains. Consequently, decision-makers are advised to carefully evaluate the diminishing returns of targeting near-complete coverage. Moreover, by identifying inflection points in the Pareto curves, planners can pinpoint "zones of opportunity" in which small additional investments yield substantial improvements in coverage, thereby guiding them away from inefficient allocation zones.

While these insights are valuable for strategic planning, the model's applicability rests on specific premises that define its current scope. We assume average vehicle ranges and driver behaviour, in which users are willing to deviate by up to a fixed threshold δ. Although this simplifies flow-capture modeling, it may not fully capture real-world variability when driver preferences are heterogeneous. Additionally, our continuous location relaxation assumes that stations can be built anywhere along an edge, and this is how we test it, limiting the location to only the segments to install is also feasible to the model without modifications besides splinting some edges and fixing some $x_a = 0$; complying with local zoning laws or power grid constraints may impose strict feasibility constraints in specific applications.

Despite these limitations, the study advances the scale of solvable problems in the literature. The introduction of 26 new large-scale instances based on the Brazilian road network can serve as benchmarks for other studies. The success of our separation procedure in handling the combinatorial explosion of sub-path constraints further indicates that this modeling approach is not limited to refuelling. And the possibility of continuous locations enables future deployments at lower infrastructure costs.

7. Conclusions

This study addressed the strategic challenge of locating alternative-fuel refuelling stations by proposing the Sub-path Flow Refuelling Location Model (SPFRLM) and the Smoothest Descent Algorithm (SDA). Motivated by the need to navigate conflicting objectives—minimizing infrastructure costs while maximizing service coverage—we developed a decision support system capable of operating at scale on large real-world road networks.

The experimental results on 26 new instances derived from the Brazilian national network validate the efficacy of this approach. We demonstrated that the SDA, particularly when coupled with the Two-Phase Hybrid (TPH) cover method, can approximate the Pareto front with high precision, capturing over 97% of the optimal hyper-volume while requiring only a fraction of the computational time of exact methods. This efficiency is crucial for decision-makers, as it enables the rapid identification of "zones of opportunity" and the assessment of marginal returns on investment without the prohibitive computational costs of exact enumeration. Ultimately, the proposed system bridges the gap between theoretical optimization and practical, data-driven infrastructure planning.

In future work, we aim to enhance the solvability of the presented formulation by investigating advanced decomposition techniques. Specifically, we intend to develop exact coverage methods based on column generation or Benders decomposition, as proposed in (Cordeau et al., 2019), to efficiently tackle more challenging instances of the FRLP. Furthermore, the separation procedure can be optimized by implementing specialized algorithms for the k single-resource constrained shortest path problem (Festa, 2015). From a methodological perspective, we plan to extend the current model to a dynamic or stochastic framework, similar to the approach proposed by (Chaieb and Ben Sassi, 2024). This will allow for the incorporation of plug-in hybrid vehicles and the simulation of heterogeneous ranges, thereby capturing the uncertainties inherent in the transition to AFVs.

CRediT authorship contribution statement

Bruno Salezze Vieira: Data curation, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft; **Glaydston Mattos Ribeiro:** Conceptualization, Data curation, Formal analysis, Validation, Supervision, Writing – review & editing; **Antônio Augusto Chaves:** Funding acquisition, Methodology, Software, Supervision, Validation, Writing – review & editing.

Data availability

The instance data is available at https://github.com/brunosalezze/frlp-instances.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

Bruno Salezze Vieira reports financial support was provided by State of Sao Paulo Research Foundation. Antonio A. Chaves reports financial support was provided by State of Sao Paulo Research Foundation. Glaydston M. Ribeiro reports financial support was provided by Carlos Chagas Filho Foundation for Research Support of Rio de Janeiro State. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This work was supported by the São Paulo Research Foundation (FAPESP) under grant 2021/09482-4. Antônio A. Chaves was supported by FAPESP under grants 2018/15417-8, 2024/08848-3, and 2022/05803-3, and Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) under grants 305557/2024-6 and 402492/2025-0. Glaydston M. Ribeiro was supported by Rio de Janeiro Research Foundation (FAPERJ) under grant E26/204.140/2024 and CNPq under grants 315694/2021-1 and 407456/2025-2.

Appendix A. Network Compression

Given a cover C that does not violate any sub-path and has an unfeasible continuous location, Algorithm 7 finds a minimum set of edges A' that requires one more RS to cover the OD pairs from W_C. It operates on a copy of the original graph G' and initiates A' with all edges from covered sub-paths (Line 1). The main loop tests each edge $a \in A'$ (Line 2) and compresses its length while saving the original value (Line 3). If the modified network generates a continuous location (Line 4), a has its length restored (Line 5), otherwise a is removed from A' (Line 7). After the main loop, the resulting A' is returned (Line 8).

Algorithm 7: Network Compression.

Data: A cover C and graph G'.
Result: Minimal set of edges A'.
1 $A' \leftarrow \{a | a \in s, \forall s \in S_w, \forall w \in W_C\}$;
2 **for** $a \in A'$ **do**
3 $\quad l_a \leftarrow |a|$; $|a| \leftarrow 0$;
4 $\quad$ **if** $ContinuousLocation(C, G')$ *is feasible* **then**
5 $\quad\quad |a| \leftarrow l_a$;
6 $\quad$ **else**
7 $\quad\quad A' \leftarrow A' \setminus \{a\}$;
8 **return** A'

Algorithm 7 is similar to linear row removal in a linear formulation to identify a minimum unfeasible system, i.e., the further removal of any other edge results in a feasible continuous location. Compressing one edge is equivalent to installing an RS at a continuous location. The resulting A' from Algorithm 7 is a set such as the addition of any RS results in a feasible continuous location, therefore $K_{A'W'} = 1 + \sum_{a \in A'} C_a$ and $W' = \{w | w \in W_C \wedge (\exists a \in A' | a \in A^w)\}$.

Taking Fig. 3 as an example for Algorithm 7, the compression of edges $\overline{ae}$ and $\overline{fd}$ still results in an infeasible continuous location, but the compression of $\overline{ce}$, $\overline{ef}$ or $\overline{fb}$ results in a feasible continuous location. Thus, the final A' is $\{\overline{ce}, \overline{ef}, \overline{fb}\}$ and $K_{A'W'} = 2$.

Computational Optimizations. This algorithm only generates one valid constraint 4 per given cover. More unique constraints are obtained by, at the end of the algorithm, picking one edge $a \in A'$, permanently compressing it ($|a| \leftarrow 0$) and checking a continuous cover, if not feasible run Algorithm 7 again to generate a different valid constraint 4, this can be repeated until the permanently compressing of some edge make a continuous location feasible.

Appendix B. Separation Problem - Continuous Location

Similar to the main separation problem (Section 4.3), the separation problem for continuous location is: given a cover C and a set of continuous locations L, for each OD pair $(o, d) \in W_C$, is there any valid path

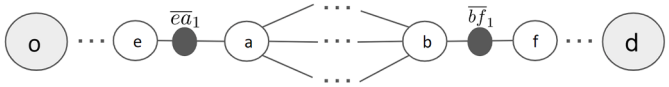

Fig. B.10. General case of a continuous location separation problem.

that contains a segment longer than R between two located RSs from L. If no, L is feasible; otherwise, a violated path segment constraint (26) is returned.

Fig. B.10 shows the general version of the problem, having $\overline{ea}_1$ on edge $(e,a) \in A$ and $\overline{bf}_1$ on edge $(b,f) \in A$. We search a candidate path p that goes from o to e, has sub-path s between a and b, with (B.1), and goes from f until d. Sub-path s must (B.1) and not pass through any edge with a located RS to violate a constraint (26).

$$|\overline{ea}_1 a| + |s| + |b\overline{bf}_1| > R \tag{B.1}$$

If such s exists, we search a s constraining p to $D^{\max}_{od}$, the path segments between (o,e) and (f,d) can always be replaced by their respective minimal path segments that do not include edges $\overline{ea}$ and $\overline{bf}$; thus, there is no need to have an explicit p leaving $|s|$ constrained by (B.2). Furthermore, as C respects all Constraints 3, $|s|$ must also be equal or smaller than R (B.3).

$$D^{min}_{oe} + |\overline{ea}| + |s| + |\overline{bf}| + D^{min}_{fd} \leq D^{max}_{od} \tag{B.2}$$

$$|s| \leq R \tag{B.3}$$

Given our network, four possible cases exist for each pair of edges and an OD pair. For edges with one or multiple RSs, the continuous location (4.3.2), x is the closest RS to a, and y is the closest RS to b, following the continuous location input.

Formulation (B.4) - (B.11), given a network G^{ab}, solves the separation problem for continuous location. It searches on sub-graph $G^{ab} = \{N^{ab}, A^{ab}\} \subseteq G$, with set of nodes $N^{ab} := \{h | h \in N,\ D^{\min}_{ah} + D^{\min}_{hb} \leq D^{\max}_{od}\}$ and set of edges $A^{od} := \{(i,j) | (i,j) \in A,\ i,j \in N^{od}\} \setminus \{\overline{a} | C_{\overline{a}} > 0\}$.

Let K^{ab}_i be constants where $K^{ab}_i = 1$ if $i = a$ or $i = b$, 0 otherwise. Additionally, $|\overline{a}|$ is the distance of edge $\overline{a}$. For binary variables, $\chi^s_{\overline{a}} = 1$ indicates that edge $\overline{a}$ is part a sub-path $s = \{\overline{a} | \chi^s_{\overline{a}} = 1\}$ between a and b. And $\gamma^s_i = 1$ means i is an intermediary node in sub-path s.

$$\sum_{a \in A^{od}_i} \chi^s_a = 2\gamma^s_i \qquad \forall i \in N^{ab} \setminus \{a,b\} \tag{B.4}$$

$$\sum_{a \in A^{od}_i} \chi^s_a = 1 \qquad \forall i \in \{a,b\} \tag{B.5}$$

$$\sum_{\overline{ij} \in A,\, i,j \in T,\, i \neq j} \chi^s_{\overline{ij}} \leq \Psi(T) - 1 \qquad \forall T \subseteq N^{ab}, \Psi(T) \geq 3 \tag{B.6}$$

$$D^{min}_{oe} + |\overline{ea}| + \sum_{\overline{a} \in A^{ab}} |\overline{a}| \chi^s_{\overline{a}} + |\overline{bj}| + D^{min}_{jd} \leq D^{max}_{od} \tag{B.7}$$

$$\sum_{\overline{a} \in A^{ab}} |\overline{a}| \chi^s_{\overline{a}} + |\overline{\overline{ea}_1 a}| + |\overline{b\overline{bf}_1}| > R \tag{B.8}$$

$$\sum_{\overline{a} \in A^{ab}} |\overline{a}| \chi^s_{\overline{a}} \leq R \tag{B.9}$$

$$\chi^s_{\overline{a}}, \in \{0,1\} \forall \overline{a} \in A^{ab} \tag{B.10}$$

$$\gamma^s_i \in \{0,1\} \forall i \in N^{ab} \tag{B.11}$$

Formulation (B.4)-(B.11) can be observed in two parts, the first one (B.4)-(B.6) searches for a sub-path $s = \{(\overline{a} | \chi^s_a = 1\}$ between a and b. And Constrains (B.7)-(B.9) mirror the conditions (B.1) - (B.3). Finally, Constraints (B.10)-(B.11) define the domains of the variables.

If Formulation (B.4) - (B.11) returns a feasible solution and assuming edges (e,a) and (b,f) are parameterized in the way as the path from o to d. The violated constraint (26) has $\pi_{jp} = D^{\min}_{oe} + |\overline{ea}| + |s| + |\overline{bf}| l_{\overline{bf}_1}$ and $\pi_{ip} = D^{\min}_{oe} + |\overline{ea}| l_{\overline{ea}_1}$.

Computational Optimizations. (i) Knowing that $D^{\min}_{ab}$ is a lower bound to $|s|$, a violation to conditions (B.1) - (B.3) eliminates the need to compute graph G^{ab} or the model. (ii) Using the upper bounds for $|\overline{xa}| = min(R, |\overline{ia}|)$ and $|\overline{by}| = min(R, |\overline{bj}|)$, graphs G^{ab} and, consequently, model (B.4) - (B.11) can be pre-computed and the values of $|\overline{xa}|$ and $|\overline{by}|$ updated on constraint (B.8) per execution.

References

Anjos, M. F., Gendron, B., & Joyce-Moniz, M. (2020). Increasing electric vehicle adoption through the optimal deployment of fast-charging stations for local and long-distance travel. *European Journal of Operational Research*, *285*(1), 263–278. https://doi.org/10.1016/j.ejor.2020.01.055

Arslan, O., Karaşan, O. E., Mahjoub, A. R., & Yaman, H. (2019). A branch-and-cut algorithm for the alternative fuel refueling station location problem with routing. *Transportation Science*, *53*(4), 1107–1125. https://doi.org/10.1287/trsc.2018.0869

Arslan, O., & Karaşan, O. E. (2016). A benders decomposition approach for the charging station location problem with plug-in hybrid electric vehicles. *Transportation Research Part B: Methodological*, *93*, 670–695. https://doi.org/10.1016/j.trb.2016.09.001

Capar, I., & Kuby, M. (2012). An efficient formulation of the flow refueling location model for alternative-fuel stations. *IIE Transactions*, *44*(8), 622–636. https://doi.org/10.1080/0740817X.2011.635175

Capar, I., Kuby, M., Leon, V. J., & Tsai, Y.-J. (2013). An arc cover–path-cover formulation and strategic analysis of alternative-fuel station locations. *European Journal of Operational Research*, *227*(1), 142–151. https://doi.org/10.1016/j.ejor.2012.11.033

Chaieb, M., & Ben Sassi, D. (2024). An advanced hyperheuristic approach for the home health care scheduling problem with time window in deterministic and uncertain environments. *Expert Systems with Applications*, *238*, 122141. https://doi.org/10.1016/j.eswa.2023.122141

Chen, T. D., Kockelman, K. M., & Khan, M. (2013). Locating electric vehicle charging stations: parking-based assignment method for seattle, washington. *Transportation Research Record*, *2385*(1), 28–36. https://doi.org/10.3141/2385-04

Chen, Z., He, F., & Yin, Y. (2016). Optimal deployment of charging lanes for electric vehicles in transportation networks. *Transportation Research Part B: Methodological*, *91*, 344–365. https://doi.org/10.1016/j.trb.2016.05.018

Chung, S. H., & Kwon, C. (2015). Multi-period planning for electric car charging station locations: a case of korean expressways. *European Journal of Operational Research*, *242*(2), 677–687. https://doi.org/10.1016/j.ejor.2014.10.029

Church, R. L., & Drezner, Z. (2022). Review of obnoxious facilities location problems. *Computers & Operations Research*, *138*, 105468. https://doi.org/10.1016/j.cor.2021.105468

Cordeau, J.-F., Furini, F., & Ljubić, I. (2019). Benders decomposition for very large scale partial set covering and maximal covering location problems. *European Journal of Operational Research*, *275*(3), 882–896. https://doi.org/10.1016/j.ejor.2018.12.021

Dantzig, G. B., & Wolfe, P. (1960). Decomposition principle for linear programs. *Operations Research*, *8*(1), 101–111. https://doi.org/10.1287/opre.8.1.101

Deb, K., & Deb, K. (2014). Multi-objective optimization. In *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques* , pp. 403–449). Boston, MA: Springer US. https://doi.org/10.1007/978-1-4614-6940-7_15

Dong, J., Liu, C., & Lin, Z. (2014). Charging infrastructure planning for promoting battery electric vehicles: an activity-based approach using multiday travel data. *Transportation Research Part C: Emerging Technologies*, *38*, 44–55. https://doi.org/10.1016/j.trc.2013.11.001

Erdoğan, S., Çapar, İ, Çapar, İ, & Nejad, M. M. (2022). Establishing a statewide electric vehicle charging station network in maryland: a corridor-based station location problem. *Socio-Economic Planning Sciences*, *79*, 101127. https://doi.org/10.1016/j.seps.2021.101127

Farahani, R. Z., Fallah, S., Ruiz, R., Hosseini, S., & Asgari, N. (2019). Or models in urban service facility location: a critical review of applications and future developments. *European Journal of Operational Research*, *276*(1), 1–27. https://doi.org/10.1016/j.ejor.2018.07.036

Festa, P. (2015). Constrained shortest path problems: state-of-the-art and recent advances. In *2015 17th international conference on transparent optical networks (ICTON)* (pp. 1–17). https://doi.org/10.1109/ICTON.2015.7193456

Greene, D. L., Ogden, J. M., & Lin, Z. (2020). Challenges in the designing, planning and deployment of hydrogen refueling infrastructure for fuel cell electric vehicles. *eTransportation*, *6*, 100086. https://doi.org/10.1016/j.etran.2020.100086

Gurobi Optimization, L. (2024). Gurobi Optimizer Reference Manual. https://www.gurobi.com.

Göpfert, P., & Bock, S. (2019). A branch&cut approach to recharging and refueling infrastructure planning. *European Journal of Operational Research*, *279*(3), 808–823. https://doi.org/10.1016/j.ejor.2019.06.031

Hodgson, M. J. (1990a). A flow-capturing location-allocation model. *Geographical Analysis*, *22*(3), 270–279. https://doi.org/10.1111/j.1538-4632.1990.tb00210.x

Hodgson, M. J. (1990b). A flow-capturing location-allocation model. *Geographical Analysis*, *22*(3), 270–279. https://doi.org/10.1111/j.1538-4632.1990.tb00210.x

Kchaou-Boujelben, M. (2021). Charging station location problem: a comprehensive review on models and solution approaches. *Transportation Research Part C: Emerging Technologies*, *132*, 103376. https://doi.org/10.1016/j.trc.2021.103376

Kim, H., Eom, M., & Kim, B.-I. (2020). Development of strategic hydrogen refueling station deployment plan for korea. *International Journal of Hydrogen Energy*, *45*(38), 19900–19911. https://doi.org/10.1016/j.ijhydene.2020.04.246

Kim, J.-G., & Kuby, M. (2012). The deviation-flow refueling location model for optimizing a network of refueling stations. *International Journal of Hydrogen Energy*, *37*(6), 5406–5420. Optimization Approaches to Hydrogen Logistics https://doi.org/10.1016/j.ijhydene.2011.08.108

Kim, J.-G., & Kuby, M. (2013). A network transformation heuristic approach for the deviation flow refueling location model. *Computers & Operations Research*, *40*(4), 1122–1131. https://doi.org/10.1016/j.cor.2012.10.021

Kınay, O. B., Gzara, F., & Alumur, S. A. (2023). Charging station location and sizing for electric vehicles under congestion. *Transportation Science*, *57*(6), 1433–1451. https://doi.org/10.1287/trsc.2021.0494

Kuby, M., & Lim, S. (2005). The flow-refueling location problem for alternative-fuel vehicles. *Socio-Economic Planning Sciences*, *39*(2), 125–145. https://doi.org/10.1016/j.seps.2004.03.001

Kuby, M., & Lim, S. (2007). Location of alternative-fuel stations using the flow-refueling location model and dispersion of candidate sites on arcs. *Networks and Spatial Economics*, *7*(2), 129–152. https://doi.org/10.1007/s11067-006-9003-6.

Kuby, M., Lines, L., Schultz, R., Xie, Z., Kim, J.-G., & Lim, S. (2009). Optimization of hydrogen stations in florida using the flow-refueling location model. *International Journal of Hydrogen Energy*, *34*(15), 6045–6064. https://doi.org/10.1016/j.ijhydene.2009.05.050

Kinay, O. B., Gzara, F., & Alumur, S. A. (2021). Full cover charging station location problem with routing. *Transportation Research Part B: Methodological*, *144*, 1–22. https://doi.org/10.1016/j.trb.2020.12.001

Lee, C., & Han, J. (2017). Benders-and-price approach for electric vehicle charging station location problem under probabilistic travel range. *Transportation Research Part B: Methodological*, *106*, 130–152. https://doi.org/10.1016/j.trb.2017.10.011

Li, X., Ma, J., Cui, J., Ghiasi, A., & Zhou, F. (2016). Design framework of large-scale one-way electric vehicle sharing systems: a continuum approximation model. *Transportation Research Part B: Methodological*, *88*, 21–45. https://doi.org/10.1016/j.trb.2016.01.014

Lim, S., & Kuby, M. (2010). Heuristic algorithms for siting alternative-fuel stations using the flow-refueling location model. *European Journal of Operational Research*, *204*(1), 51–61. https://doi.org/10.1016/j.ejor.2009.09.032

Lin, R.-H., Ye, Z.-Z., & Wu, B.-D. (2020). A review of hydrogen station location models. *International Journal of Hydrogen Energy*, *45*(39), 20176–20183. The 7th International Conference on Energy, Engineering and Environmental Engineering https://doi.org/10.1016/j.ijhydene.2019.12.035

Liu, G., & Ramakrishnan, K. G. (2001). A*prune: an algorithm for finding k shortest paths subject to multiple constraints. In *Proceedings IEEE INFOCOM 2001. conference on computer communications. twentieth annual joint conference of the IEEE computer and communications society (cat. no.01CH37213)* (pp. 743–749). (*2*). https://doi.org/10.1109/INFCOM.2001.916263

Liu, J. (2012). Electric vehicle charging infrastructure assignment and power grid impacts assessment in beijing. *Energy Policy*, *51*, 544–557. Renewable Energy in China https://doi.org/10.1016/j.enpol.2012.08.074

Majhi, R. C., Ranjitkar, P., Sheng, M., Covic, G. A., & Wilson, D. J. (2021). A systematic review of charging infrastructure location problem for electric vehicles. *Transport Reviews*, *41*(4), 432–455. https://doi.org/10.1080/01441647.2020.1854365

Mehlhorn, K., & Ziegelmann, M. (2000). Resource constrained shortest paths. In M. S. Paterson (Ed.), *Algorithms - ESA 2000* (pp. 326–337). Berlin, Heidelberg: Springer Berlin Heidelberg.

Melo, M. T., Nickel, S., & Saldanha-da Gama, F. (2009). Facility location and supply chain management – a review. *European Journal of Operational Research*, *196*(2), 401–412. https://doi.org/10.1016/j.ejor.2008.05.007

MirHassani, S. A., & Ebrazi, R. (2013). A flexible reformulation of the refueling station location problem. *Transportation Science*, *47*(4), 617–628. https://doi.org/10.1287/trsc.1120.0430

Nemhauser, G. L., & Wolsey, L. A. (1988). Integer programming and combinatorial optimization. Wiley, Chichester. GL Nemhauser, MWP Savelsbergh, GS Sigismondi (1992). Constraint Classification for Mixed Integer Programming Formulations. *COAL Bulletin*, *20*, 8–12.

Seo, S.-K., Yun, D.-Y., & Lee, C.-J. (2020). Design and optimization of a hydrogen supply chain using a centralized storage model. *Applied Energy*, *262*, 414–452. https://doi.org/10.1016/j.apenergy.2019.114452

Tu, W., Li, Q., Fang, Z., Shaw, S.-l., Zhou, B., & Chang, X. (2016). Optimizing the locations of electric taxi charging stations: a spatial–temporal demand coverage approach. *Transportation Research Part C: Emerging Technologies*, *65*, 172–189. https://doi.org/10.1016/j.trc.2015.10.004

Wang, Y.-W., & Lin, C.-C. (2009). Locating road-vehicle refueling stations. *Transportation Research Part E: Logistics and Transportation Review*, *45*(5), 821–829. https://doi.org/10.1016/j.tre.2009.03.002

Yang, J., Dong, J., & Hu, L. (2017). A data-driven optimization-based approach for siting and sizing of electric taxi charging stations. *Transportation Research Part C: Emerging Technologies*, *77*, 462–477. https://doi.org/10.1016/j.trc.2017.02.014

Zhang, Y., Liu, X., Zhang, T., & Gu, Z. (2019). Review of the electric vehicle charging station location problem. In G. Wang, M. Z. A. Bhuiyan, S. De Capitani di Vimercati, & Y. Ren (Eds.), *Dependability in sensor, cloud, and big data systems and applications* (pp. 435–445). Singapore: Springer Singapore.

Zhu, Z.-H., Gao, Z.-Y., Zheng, J.-F., & Du, H.-M. (2016). Charging station location problem of plug-in electric vehicles. *Journal of Transport Geography*, *52*, 11–22. https://doi.org/10.1016/j.jtrangeo.2016.02.002

Yildiz B., Arslan O., Karaşan O. E. 2016, A branch and price approach for routing and refueling station location model, *European Journal of Operational Research 248*, 815–826. https://doi.org/10.1016/j.ejor.2015.05.021.