Transportation Research Part E 218 (2027) 105381

Contents lists available at ScienceDirect

Transportation Research Part E

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

A supervised machine learning framework to predict the request fit for rural on-demand minibus services

Simon Mader*, Pirmin Fontaine, Stefan Voigt

Catholic University of Eichstätt-Ingolstadt, Ingolstadt School of Management (WFI) & Mathematical Institute for Machine Learning & Data Science (MIDS), Auf der Schanz 49, Ingolstadt, 85049, Germany

ARTICLE INFO

Keywords:
Transportation
Dynamic dial-a-ride problem
Machine learning
Real-time decision

ABSTRACT

On-demand minibus services offer an efficient solution for enhancing public transport in rural areas through flexible routes and schedules. Public transport providers (PTPs) aim to maximize passenger service with these minibuses, yet they encounter the challenge of a dynamic booking process where requests arrive continuously over time. PTPs have to decide immediately which requests to accept but some accepted requests may tie up the resources for future requests. An informed decision on the impact of requests could enable the PTP to serve more passengers. Therefore, the problem of predicting the request fit for rural on-demand minibus services emerges. We propose a request fit predictor (RFP) framework that leverages supervised machine learning models to predict request fit from displacement labels derived via problem-specific features. A time-dependent threshold adjusts fit requirements over the booking horizon, balancing demand and uncertainty. The modular design integrates these fit predictions with either an exact solver or a heuristic method, guaranteeing feasible real-time acceptance decisions. Tested on 26 months of real data from a German PTP, we show that the RFP improves outcomes primarily by changing which requests are accepted rather than how many: the number of accepted requests increases only marginally, while served passengers rise by 15.98% compared to the status quo and by 9.32% over a sequential solver benchmark. The RFP further sustains a stable service level across the entire booking horizon, substantially outperforming a first-come-first-served solver, and does so without excluding any bus stop from service, unlike a distance-based policy. Practitioners benefit from this paper by getting evidence that in resource-constrained rural systems, acceptance composition, service-level stability, and equitable coverage, rather than raw acceptance volume, are the primary levers for improving on-demand transport. For researchers, it offers empirical evidence on a real rural case study of where an offline-learned decision rule is competitive under sparse demand and rigid time-window constraints.

1. Introduction

Public transport providers (PTPs) face the challenge to offer affordable mobility to the community within their limited budget. This issue becomes particularly pronounced in rural areas, where lower population densities, longer travel distances, and higher operational costs per passenger, often result in limited service availability and frequency by public transport (Porru et al., 2020). The operation of conventional buses would stress the budget disproportionately in comparison to the number of served passengers (Coutinho et al., 2020). However, to promote sustainable mobility, PTPs have to explore innovative concepts to overcome these challenges and provide attractive transport options.

* Corresponding author.
E-mail address: simon.mader@ku.de (S. Mader).

https://doi.org/10.1016/j.tre.2026.105381
Received 18 March 2026; Received in revised form 10 October 2026; Accepted 5 November 2026
Available online 17 November 2026
1366-5545/© 2026 The Author(s). Published by Elsevier Ltd. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).
Open access funding enabled and organized by Projekt DEAL.

One promising approach is the implementation of on-demand bus services (Vansteenwegen et al., 2022). Unlike traditional fixed-route and fixed-schedule public transport, on-demand buses operate in response to passenger requests and dynamically adapt routes and schedules. This flexibility offers a level of convenience comparable to taxi services while maintaining the efficiency advantages of shared transport (Estrada et al., 2021). In contrast to individual taxi rides, requests are consolidated into shared journeys, reducing costs for passengers and enabling more efficient resource utilization by the PTP. In practice, such services are typically operated with small-capacity vehicles rather than large buses, we therefore use the term "on-demand minibus service". It is important to distinguish the operating mode considered in this paper from "flex-route" or deviation-based bus services, where vehicles follow a predetermined sequence of stops and flexibility arises only from selectively skipping stops or making limited detours. In contrast, the on-demand minibus system studied here does not follow a fixed stop sequence. Instead, vehicles are routed dynamically among a predefined set of stops, each of which may be visited multiple times throughout the day. Both the subset of stops to be served and the order in which they are visited are determined based on realized demand and time-window constraints.

Operationally, these systems rely on real-time data and continuous interaction between passengers and the PTP. Passengers submit requests specifying pickup and drop-off stops, number of passengers, and preferred time windows. While vehicle routes remain fixed once the service begins, the booking phase constitutes a dynamic decision problem: each incoming request must be accepted or rejected immediately, a defining characteristic of dynamic ride-sharing and ride-pooling systems (Agatz et al., 2012). This decision is complex because feasibility alone is insufficient. Accepting a request may restrict the system's ability to accommodate future, potentially more compatible requests. For example, servicing geographically remote stops during peak periods may occupy vehicles, preventing later requests from being served.

To serve as many passengers as possible within limited resources, information about the impact of the request on future requests would benefit the PTP. Naturally, if capacity were sufficiently large to serve all requests, the use of a request fit predictor would no longer be necessary. Recent routing literature also suggests incorporating information about potential future events to improve online decision-making. Anticipatory approaches can enhance solution quality but typically require substantial computational effort (Powell, 2007; Ulmer et al., 2019), limiting their suitability for real-time applications. Because booking decisions must be made instantaneously, computationally intensive optimization is impractical in live systems. Ritzinger et al. (2022) demonstrates that even advanced anticipatory heuristics often lack sufficient response speed for operational use and advocates integrating machine learning (ML). By learning patterns from historical data offline, such methods enable faster responses in online scenarios.

In this study, we consider a rural PTP aiming to maximize the expected number of served passengers under constrained resources. Requests may be placed up to 30 days in advance and as late as one hour before departure. The service area comprises a fixed network of 70 predefined bus stops distributed over approximately 100 square kilometers. Passengers can book rides from these bus stops, but the bus route is determined entirely by the composition of the incoming requests. In the rural setting considered here, vehicles have eight seats and are therefore more accurately described as minibuses. Vehicle capacity and pooling potential are closely related in dynamic dial-a-ride problems (DARPs). In dense urban settings, higher vehicle capacities can be exploited due to substantial spatial and temporal overlap of requests (Alonso-Mora et al., 2017a). In contrast, rural systems typically exhibit low, sporadic demand and tight pickup time windows, which naturally limit effective pooling (Ho et al., 2018). In our case, average vehicle occupancy remains well below the eight-seat capacity, while the rejections arise from time-window or ride-time infeasibility rather than capacity limits. This structural characteristic strongly influences the design of our solution approach.

To address the problem of dynamically accepting or rejecting requests, we develop an ML-based framework, termed request fit predictor (RFP) and use it as an empirical instrument to characterize what governs efficient and equitable request acceptance under the scarce, fixed capacity typical of rural PTPs. The RFP is designed to learn a prediction function that assesses the potential impact of a request on future requests. Assuming a static setting, we introduce the passenger-maximizing dial-a-ride problem (PM-DARP) that models the rural on-demand minibus setting with the objective to serve as many passengers as possible. The RFP is trained on a displacement label derived from repeatedly solving the static PM-DARPs on historical requests. To be more precise, a request is deemed a good fit, when it does not reduce the optimal objective value if the respective request is forced to be accepted. In contrast, a request is deemed a bad fit when its acceptance would decrease the overall objective value. By deriving features from each request that explain its label (i.e., the request fit), the RFP is learned offline based on the static PM-DARP and can then be applied online to dynamic PM-DARPs. The contributions of our paper are the following:

1. We propose a RFP framework for dynamic PM-DARPs in rural areas with tight time windows and sparse demand. The framework combines established supervised ML models with a displacement-based label and a time-dependent acceptance threshold that adapts fit requirements over the booking horizon, thereby addressing the specific characteristics of rural on-demand transport.
2. Using 26 months of real booking data from a German PTP, we show that rural on-demand systems are structurally constrained by time-window synchronization and spatial dispersion rather than vehicle capacity. As a result, anticipation is most effective at the acceptance stage, while more sophisticated online routing approaches provide only limited additional benefit.
3. By integrating the fit predictor into either an exact solver or a heuristic, the proposed framework guarantees feasible real-time routing decisions and can be incorporated into existing on-demand transport systems. Our experiments show that it maintains a stable service level throughout the booking horizon while preserving broad geographical accessibility without systematically excluding individual bus stops.

The paper is structured as follows. Section 2 introduces the problem formulation. In Section 3, the literature review places the problem into its theoretical context. Section 4 describes the RFP framework. Section 5 presents the experimental design including the case study. Section 6 presents the numerical results and draws on the implications. The paper concludes in Section 7.

2. Problem formulation

In this section, we introduce the problem of predicting the request fit for rural on-demand bus services modeled as a dynamic PM-DARP. First, we introduce the event-based mixed-integer program (MIP) of the static problem formulation, in Section 2.1. Section 2.2 draws on the static problem formulation to derive the dynamic PM-DARP. Third, we motivate the decision impact and the need for an informed decision through an illustrative example in Section 2.3. The notation can be found in Appendix A.

2.1. Static problem

Typically, the static DARP aims to minimize the distance driven (Cordeau and Laporte, 2003). However, the objective for PTPs shifts toward maximizing the number of served passengers. This objective reflects the primary policy goal of improving accessibility and coverage rather than revenue optimization in low-density rural areas (Liyanage and Dia, 2025). Therefore, we use the PM-DARP where not all available requests can be served and decisions must be made to determine the set of transported passengers. Unlike other selective DARP variants, the possible profits (Parragh et al., 2015) and the passenger preference on different offers and prices (Dong et al., 2022; Azadeh et al., 2022) are neglected for the PTP. All passengers are equally valuable for the PTP and the only goal is to serve as many passengers as possible, as fares are fixed and revenue management is not part of the public transport setting considered here (for price-based demand control, see for example Fleckenstein et al., 2023). Compared to selective DARP variants that maximize the number of served requests (Riedler and Raidl, 2018; Rist and Forbes, 2022), our problem maximizes the number of served passengers, thereby accounting for requests that may include multiple passengers.

Each request $r_i \in R = \{r_1, \dots, r_n\}$ consists of a pickup location i^+, a corresponding drop-off location i^-, the number of passengers q_i, and the preferred pickup time t_i. The homogeneous fleet of available vehicles is denoted by the set K, and each vehicle in K is responsible for serving one or more of the requests in R. All vehicles share identical capacity, availability, operating hours, and operational constraints. The full set of locations is given by $J = P \cup D \cup \{0^+, 0^-\}$ where $P = \{1^+, \dots, n^+\}$ denotes the set of pickup locations, $D = \{1^-, \dots, n^-\}$ represents the set of drop-off locations, and $\{0^+, 0^-\}$ corresponds to the depot locations used as the start and end points for each vehicle route. Each location $j \in J$ is associated with a time window $[e_j, l_j]$ during which service must begin. For every request $r_i \in R$, the pickup location i^+ is assigned a time window $[e_{i^+}, l_{i^+}]$ based on the customer's preferred pickup time t_i. The drop-off location i^- is assigned a time window $[e_{i^-}, l_{i^-}]$, which is calculated by adding the direct travel time t_{i^+,i^-} to the pickup time window $[e_{i^+}, l_{i^+}]$. In addition to time windows, each request has a service time $s_i \geq 0$ required at both pickup and drop-off locations. To ensure passenger comfort and limit detours, a maximum ride time constraint is introduced. For each request i, the maximum allowed ride time is defined as $L_i = t_{i^+,i^-}(1 + \varepsilon)$ where ε is a parameter representing the maximum additional ride time factor.

The static PM-DARP can be formulated in a straight-forward manner on a complete graph that connects all nodes in J with arcs associated with the distance between nodes (i.e., location-based formulation, Parragh et al., 2008). We however, use the event-based formulation of Gaul et al. (2022, 2025) and similarly defined by Anzenhofer et al. (2025). This formulation is particularly well suited to the rural on-demand minibus setting studied in this paper, which is characterized by low request volumes, tight time windows, limited pooling opportunities, and small-capacity minibuses. Under these conditions, infeasible event transitions can be effectively eliminated during preprocessing, leading to compact event-based graphs with fewer variables and tighter linear relaxations. While event-based formulations are known to scale poorly in settings with large vehicle capacities or extensive temporal overlap between requests (Stallhofer and Parragh, 2025), such conditions are not representative of the rural public transport context considered here. In preliminary experiments, we alternatively generated displacement labels using a conventional location-based PM-DARP formulation. Although this approach led to substantially longer runtimes and frequently failed to prove optimality within reasonable time limits, the resulting improvements in accepted requests were comparable. This indicates that the proposed learning framework is robust with respect to the underlying optimization model used for label generation.

In the event-based formulation, the problem is modeled on an event-based graph $G = (V, A)$ where nodes V represent the set of events and arcs A the transition between events. An event refers to a pickup or a drop-off of some requests and contains information about the current location and all requests on board of the corresponding vehicle k. Vehicle indices are not modeled explicitly. Instead, the event-based formulation implicitly represents vehicles via event flows, with the fleet size enforced through a depot outflow constraint, i.e., at most $|K|$ vehicles can leave the depot. The event is represented by a Q-tuple $(v_1, \dots, v_Q)$, where Q denotes the vehicle capacity in passengers. Since each request contains at least one passenger, Q also provides an upper bound on the number of requests that can be simultaneously on board and therefore on the dimensionality of an event. Each component $v_1, \dots, v_Q$ refers to a request $r_i \in R$. Importantly, only capacity-feasible events are included in V, i.e., the total number of passengers associated with the requests on board must satisfy $\sum_{i \in R(v)} q_i \leq Q$. The first component v_1 encodes the vehicle's current location (the request whose pickup or drop-off just occurred) while the remaining components $v_2, \dots, v_Q$ identify the requests currently on board, each referring to a request $r_i \in R$ or being empty if fewer than $Q - 1$ requests are on board. A given combination of location and on-board requests thus defines one event; enumerating all feasible such combinations yields the event set V, so that Q bounds the dimensionality of each event and thereby the size of V, whereas V is the collection of all realizable events. The set of events V consists of the depot event V_0, pickup events V_{i^+} and dropoff-events V_{i^-}. Arcs represent transitions between events with the changes in requests on board. The arc set A is divided into six disjoint subsets, each corresponding to a distinct type of feasible transition between depot, pickup, and drop-off events: pickup-to-drop-off, pickup-to-pickup, drop-off-to-pickup, drop-off-to-drop-off, drop-off-to-depot, and depot-to-pickup transitions. For the formal definition of these subsets we refer to Gaul et al. (2022). The event-based graph implicitly enforces capacity, precedence, and pairing constraints and excludes infeasible transitions by construction. Each event $v \in V$ has a set of incoming arcs

$\lambda^{\text{in}}(v)$ and outgoing arcs $\lambda^{\text{out}}(v)$. The distance between two events, v and w, is analogous to the distance between their corresponding locations $i, j \in J$, i.e., the travel distance associated with the arc $a = (v, w) \in A$ is defined as $c_a = c_{ij}$ and the corresponding travel time t_{ij} is obtained from this distance and the assumed average travel speed. We define the scalar cost factor γ to weigh the routing distance in the objective.

For each request $i \in R$, the binary decision variable $y_i \in \{0, 1\}$ indicates whether request i is accepted or rejected. For each arc $a \in A$ in the event-based graph, the binary decision variable $x_a \in \{0, 1\}$ equals 1 if arc a is used in the route. For every location $j \in J$, the continuous non-negative decision variable $B_j \geq 0$ specifies the start time of service at location j.

Model PM-DARP

The model formulation is given as follows:

$$\max \sum_{i \in R} y_i \, q_i \; - \; \gamma \sum_{a \in A} c_a \, x_a \tag{1}$$

s.t.

$$\sum_{a \in \lambda^{\text{in}}(v):\, v \in V_{i+}} x_a = y_i \qquad \forall \, i \in R \tag{2}$$

$$\sum_{a \in \lambda^{\text{in}}(v)} x_a - \sum_{a \in \lambda^{\text{out}}(v)} x_a = 0 \qquad \forall \, v \in V \tag{3}$$

$$\sum_{a \in \lambda^{\text{out}}\left((0,\ldots,0)\right)} x_a \leq \qquad |K| \tag{4}$$

$$e_j \leq B_j \leq l_j \qquad \forall \, j \in J \tag{5}$$

$$B_{i^-} - B_{i^+} - s_i \leq L_i \qquad \forall \, i \in R \tag{6}$$

$$B_j \geq B_i + s_i + t_{ij} - M_{ij}\left(1 - \sum_{v,w \in V \,:\, v_1 = i \wedge w_1 = j} x_{(v,w)}\right) \qquad \forall \, i, j \in J \tag{7}$$

$$B_j \geq 0 \qquad \forall \, j \in J \tag{8}$$

$$x_a \in \{0, 1\} \qquad \forall \, a \in A \tag{9}$$

$$y_i \in \{0, 1\} \qquad \forall \, i \in R \tag{10}$$

The objective function (1) of the PM-DARP maximizes the number of served passengers q_i which are accepted by the decision variable y_i. The objective function is penalized by adding routing costs c_a associated with the selected event arcs x_a, scaled by a cost factor γ. By keeping this factor small, we prioritize passenger maximization, while also differentiating between solutions serving the same number of passengers. Constraints (2) link both decision variables y_i and x_a. For the remainder of the constraints, we follow the event-based DARP formulation of Gaul et al. (2025). Constraints (3) ensure event flow conservation, meaning that every transition leading to an event is matched by a corresponding transition from that event. The maximum number of vehicles used is defined by Constraint (4). Constraints (5) ensure that each time window for every location is respected. The time the passenger spends in the vehicle is limited by the maximum passenger ride time (6). Constraints (7) enforce sequential time flow: if the vehicle moves directly from location i to location j, service at j cannot start before service at i is completed and the travel time t_{ij} has elapsed. The summation indicates whether this location transition is used as event transition (v, w); if not, the big-M constant M_{ij} deactivates the constraint. Constraints (8)–(10) specify the decision variables.

2.2. Dynamic problem

In the dynamic problem, the objective and constraints remain the same as in the static PM-DARP. However, the set of requests $R = \{r_1, \ldots, r_n\}$, arrive continuously over time. The PTP faces the challenge of determining immediately whether to accept ($y_i = 1$) or reject ($y_i = 0$) a request r_i with only partial knowledge of the set of requests. The dynamic PM-DARP could be solved optimally when all requests are known in advance, i.e., assuming the PTP has perfect information (PI) about all future requests using the static formulation in the previous section. However, since requests enter the system sequentially, the dynamic DARP requires a formulation based on expected values. Typically, dynamic routing problems are modeled as Markov Decision Processes (MDP, see e.g., Ulmer et al., 2020). In this formulation, a solution is given by a policy π, specifying in every state whether to accept ($y_i = 1$) or reject ($y_i = 0$) an incoming request. An optimal policy π^* maximizes the expected total number of passengers served over the entire booking horizon, taking into account the uncertainty about future requests. Based on the current solution S which contains all of the accepted requests up to that point, the PTP has to decide in real-time if the request r_i can still be served. To determine the feasibility of the request r_i, the following constraints have to be checked:

1. **Capacity constraints** ensure that the total number of passengers on board of vehicle k does not exceed its capacity Q, i.e., requests with multiple passengers consume q_i seats.
2. **Precedence constraints** ensure that each passenger pickup i^+ precedes their corresponding drop-off i^-, and that both events are served by the same vehicle k.

3. **Maximum passenger ride time constraints** limit the duration each passenger spends onboard the vehicle to a predefined maximum ride time L_i.
4. **Maximum tour length** restrict the total travel duration of each vehicle k.
5. **Time windows constraints** require that pickups and drop-offs are performed within acceptable predefined time intervals for both vehicles and passengers.

Checking only for feasibility neglects the impact of request r_i on accepting or rejecting future requests. Therefore, we discuss the impact of this decision in the following section.

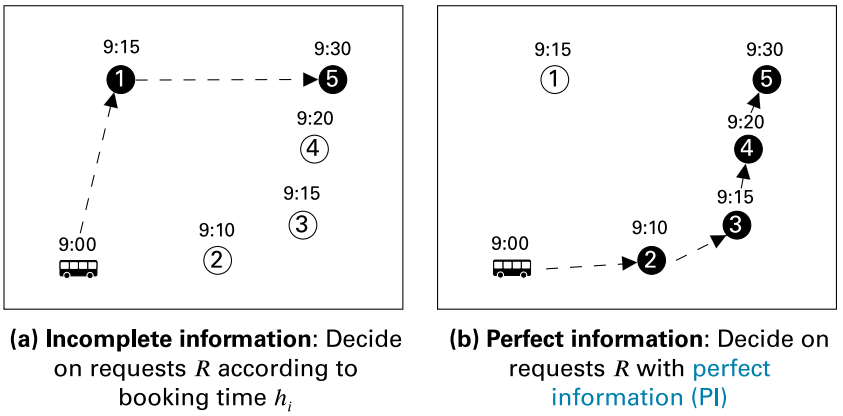
2.3. Decision impact

Accepting requests without considering the request fit may lead to suboptimal resource allocation. To illustrate the significance of the booking slot allocation, Fig. 1 presents an example of five requests $R = \{r_1, r_2, r_3, r_4, r_5\}$ with corresponding booking times during the booking horizon H with $h_1 = 7:00$, $h_2 = 7:30$, $h_3 = 7:45$, $h_4 = 7:50$, and $h_5 = 8:00$ and one passenger per request. These passengers indicate their preferred pickup times as $t_1 = 9:15$, $t_2 = 9:10$, $t_3 = 9:15$, $t_4 = 9:20$, and $t_5 = 9:30$, respectively. The example is simplified (neglecting the drop-off) to focus on the decision impact of the request acceptance.

(a) **Decisions based on incomplete information**: The PTP accepts the requests $y_i = 1$ based on the chronological order of the booking time h_i. According to this mechanism, the first request (with h_1) is accepted since all constraints are fulfilled and the route is feasible. However, requests two to four are rejected ($y_2 = 0$, $y_3 = 0$ and $y_4 = 0$) due to their distant locations and proximate preferred pickup times t_2, t_3 and t_4 relative to the already accepted first request. Consequently, only the fifth request gets a booking slot, resulting in two passengers being served.

(b) **Decisions based on perfect information**: If the PTP had complete knowledge of all incoming requests, it could make optimal decisions regarding which requests to accept by solving the static version of the problem. With this PI, the system would have rejected the first request ($y_1 = 0$) and accepted all the other requests ($y_2 = 1$, $y_3 = 1$, $y_4 = 1$ and $y_5 = 1$). This specific combination of requests not only fulfills the feasibility constraints, but also represents the best acceptance/rejection decisions for all available requests. Knowing all requests in advance allows for the best possible allocation and results in four served passengers.

r_i	h_i	t_i
1	7:00	9:15
2	7:30	9:10
3	7:45	9:15
4	7:50	9:20
5	8:00	9:30



(a) Incomplete information: Decide on requests R according to booking time h_i

(b) Perfect information: Decide on requests R with perfect information (PI)

Fig. 1. Comparison of (a) Incomplete and (b) Perfect information.

We define a request as a good fit if its presence in a solution does not reduce the objective value compared to the optimal solution. In our example, accepting r_1 led to the rejection of three other passengers (i.e., two passengers were displaced), while the acceptance of other requests displaces no passengers.

3. Literature review

Shared mobility and ride-pooling services have emerged as a key instrument for improving vehicle utilization by pooling multiple passenger requests into shared trips. This operational challenge has been studied under various labels, including dynamic ride-sharing, ride-pooling, and on-demand mobility (Vansteenwegen et al., 2022). Early work emphasizes the need for fast online matching and routing decisions to enable short-notice trip coordination (Agatz et al., 2012), while more recent research incorporates anticipatory elements such as approximation algorithms or reinforcement learning (RL) (Yu and Shen, 2019; Luo et al., 2023; Xie et al., 2023). Despite differences in terminology and application context from ride-sharing to platform-operated microtransit, these problems can generally be modeled as variants of the dynamic DARP.

This literature review locates the problem within the DARP context in Section 3.1. In Section 3.2, routing problems and solution approaches that incorporate anticipatory information are presented. Section 3.3 discusses real-time decision making in routing algorithms and how fast approaches were designed. In Section 3.4, we analyze existing ML algorithms that exploit information to make better decisions in a dynamic routing setting. Finally, Section 3.5 summarizes the literature and highlights the research gap.

3.1. Dynamic dial-a-ride problems

The DARP is a vehicle routing variant with pickup and delivery. The most common constraints relate to vehicle capacities, time windows and the maximum ride time the passenger spends in the vehicle (Cordeau and Laporte, 2003). For a summary of DARP models and algorithms, we refer the reader to Cordeau and Laporte (2007). The most recent literature overviews on DARPs by Molenbruch et al. (2017) and Ho et al. (2018) distinguish two main versions of the problem: Static and dynamic DARPs. In the static version, all passenger requests are known in advance allowing to build the solution before the execution; while in the dynamic version, some or all passenger requests arrive sequentially and the solution is adjusted with the incoming requests. In our problem, we deal with a dynamic DARP because all passenger requests arrive in real-time in the booking system and the PTP has to decide immediately if requests are accepted or rejected.

The literature on dynamic DARPs neglects exact solutions as they are not capable of providing timely solutions (Ho et al., 2018). Even metaheuristic approaches that provide high-quality solutions such as parallel tabu search (Attanasio et al., 2004) or hybrid approaches that combine tabu search and constraint programming to accelerate the route evaluation (Berbeglia et al., 2012) lack the response speed required in a real-time booking process. The insertion heuristic, as introduced by Jaw et al. (1986), lays the groundwork for modern rapid methods. This heuristic evaluates the feasibility of a new request with the existing route and adjusts the solution accordingly, either accepting or rejecting the request. Coslovich et al. (2006) create a feasible neighborhood of the current route in a first offline step to react faster in an online setting when a new request needs to be inserted in the current route. In some cases, the request cannot be inserted in the current solution. Therefore, Luo and Schonfeld (2011) test reinsertion techniques to insert the rejected requests by moving other requests that are considered similar. Comparing the immediate insertion with a rolling-horizon insertion that reschedules close requests to a later time results in fewer operating vehicles. Vallée et al. (2020) improve the immediate insertion heuristic by several destroy and repair operators and test a graph-based heuristic with intensive exploration of the neighborhood. The proposed reinsertion methods result in more transported passengers. These improvements of the online insertion heuristic are fast and result in feasible solutions but are not capable of evaluating the fit of the incoming request. In contrast, anticipatory algorithms incorporate advanced information when building the solution.

3.2. Anticipatory routing algorithms

In dynamic routing, stochastic information is often used to anticipate future requests (Soeffker et al., 2022). Voccia et al. (2019) integrate scenario sampling into dynamic programming for same-day delivery, finding that combining sampling with delay offers is beneficial. Ulmer et al. (2019) combine offline value function approximations with online rollout to anticipate requests in space and time, later applying buffer and postponement strategies in meal delivery to reduce delays (Ulmer et al., 2021). While these approaches successfully integrate anticipation, on-demand minibus settings exhibit different control characteristics. In rural contexts, services operate with fixed fleets, leaving no capacity to redeploy vehicles in response to demand fluctuations. As a result, acceptance decisions have a disproportionate and irreversible impact on future requests, making resource allocation the central challenge.

In the context of DARPs, Alonso-Mora et al. (2017b) incorporate artificial requests into their routing that represent an estimated mobility demand. The service area is divided into defined zones and routes are partially pre-planned around areas with historically high demand and then adjusted according to the incoming requests. In our problem, the demand is spatially and temporally heterogeneous, and passengers have complete freedom in their booking options, making the approach of partially predefined routes impractical for our context. Van Engelen et al. (2018) develop a more dynamic approach: They enhance their online insertion algorithm with a demand forecast that identifies which vehicle is best suited to serve the passenger. Their results show that the rejection rate can be decreased by forecasting the demand with a reasonable increase in driving distance. Additionally, rerouting empty vehicles to anticipated locations can reduce the rejection rate, even if it means vehicles travel longer distances. Regarding our PTP, the volume of requests is high compared to the number of vehicles. Given the limited resources, we cannot rely on extending travel distances as an adjustment. Therefore, we focus on strategies to decrease rejection rates.

Schilde et al. (2011) analyze hospital transports with the main decision to wait at specific locations because the return back home of the patient is anticipated. They develop a variable neighborhood search with sample scenarios and obtained the best results by single scenarios with requests in the near future. Ritzinger et al. (2022) build on the work of Schilde et al. (2011) and develop a stochastic variable neighborhood search that evaluates sampled scenarios of several possible future requests and inserts the original request in the best insertion position. They compare this algorithm against two waiting strategies that determine waiting locations based on (1) a stochastic model incorporating demand forecasts and (2) historical demand data. The best algorithm simply uses historical information for the waiting strategies. The application of the stochastic variable neighborhood search is also too time-consuming for the online setting and Ritzinger et al. (2022) propose further research to apply ML algorithms to incorporate the historical information in the decision.

3.3. Real-time decision making in routing problems

Most routing algorithms for dynamic problems benefit from anticipating future demand with scenario-sampling or dynamic programming approaches. But these approaches are too time-consuming for our real-time application. Furthermore, the demand in rural areas is low and sporadic. Therefore, constructing suitable distributions is challenging. The few approaches that consider real-time

decision making in the dynamic routing context neglect reporting computation times (Ichoua et al., 2006; Ulmer, 2020; Ulmer and Thomas, 2020). Even fewer of these real-time decision making approaches report computation times (Klapp et al., 2020; Zhang et al., 2023; Pina-Pardo et al., 2024).

Klapp et al. (2020) study an e-commerce platform that has to immediately react to customer requests. Dispatching the orders to a single truck is done in discrete moments, termed as dispatch waves. Klapp et al. (2020) design a proactive policy that rolls out an optimal a priori policy but takes more than 18 seconds on average for the acceptance decision. To make their approach applicable for real-time decisions, they integrate a metaheuristic with slightly higher costs per request (on average 2%) but computation times of around one second per request. Zhang et al. (2023) consider real-time decision problems in same day delivery with stochastic customers by approximating the reward-to-go with a knapsack-based approach. On their small instances with two vehicles the online acceptance of their potential-based approximation meets computation times of 0.4 to 0.6 seconds per request. The approach of Zhang et al. (2023) scales better than roll out algorithms with sampled paths that already need more than three seconds for such small instances. Pina-Pardo et al. (2024) study a dynamic routing problem of a single truck with drone resupply. By incorporating a cheapest insertion heuristic, Pina-Pardo et al. (2024) ensure real-time decisions with a maximum of 0.13 seconds per request.

3.4. Machine learning enhanced algorithms

Some studies show that historical information can be beneficial for fast online decisions. Utilizing ML models to extract insights from previous experiences is considered a promising approach (Ritzinger et al., 2022). Recent surveys distinguish between approaches that learn control policies directly or predict-then-optimize approaches, where ML provides decision-relevant predictions that are embedded into optimization procedures (Sadana et al., 2025). For instance, Ghiani et al. (2022) start with a base policy for a pickup and delivery problem with time windows and adjust the policy's parameters according to the incoming requests using a trained supervised ML approach. Larsen et al. (2022) estimate the value under data uncertainty in load planning for rail transportation with supervised ML models. They solve the operational bin packing problem with PI to make fast approximated decisions about accepting or rejecting an order. Recent work has increasingly focused on learning-augmented optimization, where ML models guide classical combinatorial algorithms rather than replacing them entirely (Baty et al., 2024). For example, Jungel et al. (2026) propose a learning-based framework for acceptance, dispatching, and repositioning decisions in large-scale, high-density urban fleets. In contrast, our rural setting features limited fleets and little scope for repositioning, making feasibility and real-time responsiveness critical.

Van der Hagen et al. (2024) apply supervised ML models in time slot management for a grocery retailer. Drawing on solutions from vehicle routing problems with time windows, they learned the feasibility check to offer time slots to passengers in an online booking system. Similar to our study, the speed of the algorithm is of particular importance. In addition to the feasibility check, we also predict the request fit to maximize the expected number of served requests. Van der Hagen et al. (2024) address a vehicle routing problem with time windows where most of the requests (especially the initial ones) are feasible. The feasibility in our DARP problem depends on the state of the minibus and the incoming requests that have pickup and drop-off locations and times. Moreover, the PTP has to guarantee feasibility to fulfill the service promise and not scare off future passengers.

The alternative branch of this taxonomy learns control policies directly through experience. Chen et al. (2022) demonstrate that deep Q-learning yields strong accept/reject policies for same-day delivery with vehicles and drones, and comparable results have been reported for order dispatching and repositioning in dense urban fleets. These settings share two properties that our rural application lacks: a high-volume, spatially concentrated request stream from which a policy can be learned by repeated interaction, and a comparatively unconstrained action space, since delivery requests impose no passenger pickup time windows and no precedence coupling between pickup and drop-off. Hildebrandt et al. (2026) make this limitation explicit, showing that RL evaluates reward and value functions efficiently but struggles to search the constrained decision spaces that characterize most planning problems. In the PM-DARP considered here, time windows are the binding constraint: they couple pickup and drop-off in time and make feasibility dependent on the entire committed schedule rather than on the current request alone, so the admissible action set is narrow and state-dependent at every decision epoch. Appendix H confirms this empirically: a proximal policy optimization (PPO) and a value-based deep Q-learning (DQN) agent, both observing strictly more information than our threshold, remain marginally above the service level of accepting every feasible request. We therefore adopt an imitation-based predict-then-optimize design in which the anticipatory signal is learned offline from PI solutions and feasibility is enforced by the routing component at decision time.

3.5. Research gap

Dynamic DARP solution approaches typically rely on fast insertion heuristics to maintain feasibility. However, these methods offer no mechanism to assess whether a new request truly "fits" without violating capacity, time window, or precedence constraints. Anticipatory approaches address this gap and are well established, from scenario sampling and rollout policies (Ritzinger et al., 2022; Klapp et al., 2020) to policies learned directly through experience in high-density settings (Chen et al., 2022; Jungel et al., 2026). Their online cost is substantial, however, whereas passengers in a booking system expect an immediate response. The rural setting amplifies this: demand is sparse and sporadic, so sampled future requests frequently fail to materialize, and tight pickup time windows, precedence, and fixed capacity render the admissible action space narrow and state-dependent, which is precisely where RL is known to struggle (Hildebrandt et al., 2026), as our own experiments with two RL agents confirm (Appendix H). What is missing is therefore not another anticipatory mechanism, but a way to retain anticipatory foresight without online scenario sampling while guaranteeing feasibility in real time. Pure ML approaches achieve the required speed in home delivery (Van der Hagen et al., 2024) but cannot

guarantee feasibility in our minibus context. We address this gap by encoding the anticipatory signal offline into a displacement label and enforcing feasibility through the routing component at decision time.

4. Solution method

This section introduces the proposed solution method. Section 4.1 first presents the RFP pipeline, followed by the ML training procedure in Section 4.2. Finally, Section 4.3 describes the application to the dynamic PM-DARP.

4.1. Request fit predictor pipeline

Fig. 2 illustrates the pipeline that derives information from solving static PM-DARPs with historical requests, which is then used for solving the dynamic PM-DARP. The training and application of ML models enables the translation of the PI from the static problem to the dynamic application.

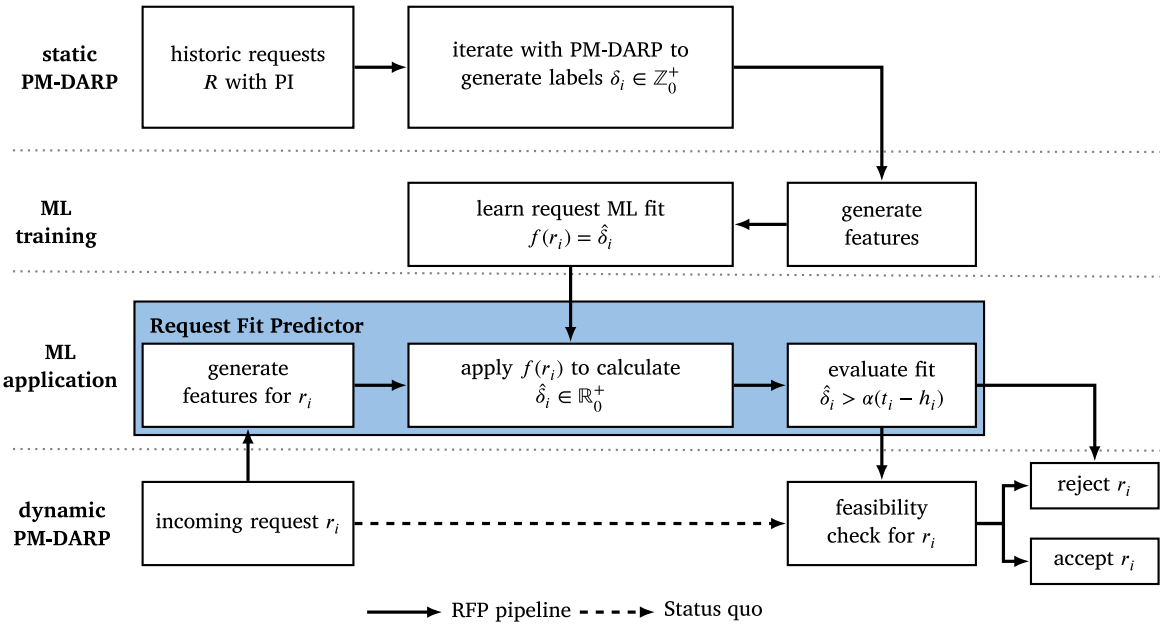


Fig. 2. RFP Pipeline: Learning from the static for the dynamic PM-DARP.

The static PM-DARP is solved using the PI of historical requests R (see upper left of Fig. 2). We compute for each request r_i the displacement label δ_i, which is used as the target label. To train the ML models, features are generated from the historical requests R that explain the request fit δ_i. To integrate the PI knowledge into the dynamic process, we introduce the ML model in form of the prediction function $f(r_i)$, that takes the request r_i as input. This function projects the potential impact of that request on future route efficiencies. Since δ_i quantifies the deviation from the optimal solution when request r_i is included, a lower value indicates a better fit. The learned prediction function for the fit of request r_i is then applied in the dynamic PM-DARP.

In the dynamic PM-DARP (lower part of Fig. 2), instead of conducting only the feasibility check and accept or reject the incoming request r_i, the RFP is applied. First, the features for the request r_i are generated. Then, the learned function $f(r_i)$ is applied on these features to calculate the request fit $\hat{\delta}_i \in \mathbb{R}_0^+$. This output represents the potential displacement (i.e., fit) of the request. To account for the diminishing uncertainty about future requests over the booking horizon, we employ a time-dependent threshold $\alpha(t_i - h_i)$. This design choice avoids the additional complexity of learning a fully adaptive acceptance policy, which would require training data spanning a large variety of arrival sequences and dynamic system states. The request r_i is either directly rejected (in case $\hat{\delta}_i$ is above $\alpha(t_i - h_i)$) or checked for feasibility in the solution. In the latter case, if r_i is also found to be feasible, it is accepted and inserted into the solution. Acceptance decisions are irrevocable. Once a request r_i is accepted, the corresponding decision variable y_i is fixed to one in all subsequent feasibility checks or re-optimization steps. Rejected requests are removed from the system and cannot be reconsidered. Consequently, any re-optimization is performed conditional on the set of previously committed decisions.

4.2. Machine learning training

In this section, the components of the ML training are outlined. First, we describe the generation of the labels followed by the calculation of the features. The features have to be generated for the training and application phase. Moreover, we describe the ML models and the time-dependent threshold.

4.2.1. Label generation

In order to derive the labels, we first solve the static PM-DARP under PI (see Section 2.1) to generate an optimal solution with the value F^{PI*}. In this solution, each request r_i is assigned a binary decision variable $y_i \in \{0, 1\}$ indicating whether it is accepted. To quantify the fit of each request, we then compute a displacement label. For each request r_i, we force its inclusion in the solution and obtain the corresponding objective value F_i^{PI} by re-solving the static PM-DARP. The displacement label is then defined on the served passenger count only, as $\delta_i = P^{PI*} - P_i^{PI} \in \mathbb{Z}_0^+$, where P^{PI*} and P_i^{PI} denote the number of passengers served in the unconstrained optimal solution and in the solution with r_i forced into it, respectively. The distance term in the objective (see Section 2.1) enters the optimization solely as a symmetry-breaking penalty, ensuring that the solver deterministically selects the shortest route among solutions serving an identical number of passengers; it does not enter the label. The labels are therefore non-negative integers throughout, consistent with the count-data models applied in Section 4.2.3. A displacement of $\delta_i = 0$ indicates that including r_i does not worsen the optimal objective, signifying an ideal fit. Conversely, a positive displacement (e.g., $\delta_1 = 2$ in Fig. 1) means that accommodating r_1 forces the rejection of two other requests, thus reflecting a less favorable fit. These displacement values serve as labels for training the ML models.

4.2.2. Feature generation

Features are generated in the training phase and in the application process. The information from the incoming request $r_i = \{i^+, i^-, q_i, t_i, h_i\}$ is the basis for the feature preparation. We prioritize static and geography-based features to ensure generality across system states. Other dynamic conditions are captured implicitly via feasibility checks. Based on this request information, we generate 34 features: 12 geographical, 17 time, 2 capacity features, and 3 aggregated demand features. We describe the resulting set of features below. Table B.1 in Appendix B presents the features.

Geographical features are derived from the latitude and longitude of each pickup i^+ and drop-off i^- to define trip origin and destination. From these coordinates, we compute trip distance as a proxy for travel time and resource use. To capture spatial context, we add local density measures: (1) The gravity distance is defined as the straight line distance from the pickup to a central network point and indicates remoteness from the service core, which is especially relevant in rural areas. (2) Distance to the nearest bus stop highlights potentially underserved locations. (3) Average distances to the five and ten nearest bus stops reflect local and regional density, respectively, based on a k-means clustering of 70 stops into ten clusters. The five smaller clusters each contain, on average, around five stops, while the five larger clusters have about ten bus stops each.

Time features capture the temporal characteristics of ride requests. We start by recording the pickup t_{i^+} and drop-off times t_{i^-}, measured in minutes, which allow us to compute the estimated ride time as the difference between these two timestamps. To simplify the analysis, the pickup time is rounded to the nearest full hour, referred to as the trip hour. Additionally, ride requests are segmented into three parts of the day: morning (before noon), noon (between noon and 5 pm), and evening (after 5 pm), to capture different demand patterns. The periods are differentiated to account for the varying service hours: 18 hours on weekdays, 14 hours on Saturdays, and 12 hours on Sundays. Moreover, since drivers can operate for a maximum of four consecutive hours before they are required to take a break, we segment each service day into four operational slots of four hours each. To achieve even finer temporal resolution, each four-hour slot is subdivided into four quartiles, each representing one hour.

Capacity features consist of (1) the number of passengers q_i for each request r_i. This feature provides insight into the demand for each trip, highlighting whether requests involve individuals or groups; and (2) the booked distance per passenger. By normalizing the distance by the number of passengers, we obtain a more precise measure of resource utilization per traveler.

Aggregated demand features capture the spatial and temporal patterns of requests within the network by leveraging the fixed bus stop system, which is organized into ten clusters. For each cluster, we compute demand metrics at three levels of granularity: daily, within four-hour slots (aligned with driver tours), and hourly. These metrics are derived as the mean number of requests for each day of the week.

4.2.3. Machine learning models

We model the prediction of the request fit as a prediction task with count data (McCullagh, 2019) as the probable displacement and can draw on the well-established literature of ML models (Hastie et al., 2009). Bengio et al. (2021) provide an extensive overview of ML models with application in combinatorial optimization problems. According to their taxonomy, our supervised ML models approximate decisions learned on ground truth labels (i.e., imitation learning) instead of exploring new policies through experience (i.e., experience learning) (Bengio et al., 2021). In our approach, we train the ML models on the PI solution of the served requests to imitate the decisions in a static PM-DARP for the dynamic setting. We focus on three prominent ML models: Random Forest, XGBoost, and Neural Networks.

Random Forest is an ensemble learning model that constructs multiple decision trees at training time and outputs the mode of the class of the individual trees (Breiman, 2001). By averaging multiple decision trees the algorithm is less prone to overfitting and often produces a reasonable model with the default hyperparameters (Probst et al., 2019). Random Forest is used as the standard ML benchmark in our study because of its general good performance across different scenarios (Fernández-Delgado et al., 2014).

XGBoost is the short form for eXtreme Gradient Boosting, a highly efficient and scalable implementation of gradient boosting machines (Chen and Guestrin, 2016). This algorithm works by sequentially building trees, where each new tree learns from the errors made by the previous ones, effectively improving the model with each step (Friedman, 2001). One of the key advantages of XGBoost is its ability to handle a wide range of data types, distributions, and relationships, making it extremely versatile. The model also includes regularizations, which helps reduce overfitting and provides a rich set of hyperparameters that can be finely tuned to enhance model performance (Chen and Guestrin, 2016). The output can explicitly be trained on Poisson-distributed count data and

assessed with the negative log-likelihood criterion. Due to these characteristics, XGBoost is selected as an advanced model in our study, known for its exceptional performance in a variety of ML tasks and competitions (Nielsen, 2016).

Neural Networks, particularly deep learning, consist of hierarchical layers of neurons that apply non-linear transformations to input data (LeCun et al., 2015). Learnable parameters and activation functions shape these transformations, enabling successive layers to capture increasingly complex patterns (Sutskever et al., 2013). In our approach, the final layer uses an exponential activation to ensure strictly positive displacements (Goodfellow et al., 2016). Neural Networks perform well on high-dimensional data, as shown in image recognition successes like ImageNet (Krizhevsky et al., 2012), but require large datasets to avoid overfitting (Zhang et al., 2021). Although our dataset is smaller, we include a Neural Network as a benchmark for testing complex model performance.

4.2.4. Time-dependent threshold

To effectively account for the inherent variability in passenger demand and routing conditions, our framework adopts a time-dependent threshold for request acceptance, denoted as $\alpha(t_i - h_i)$, where the threshold is adjusted based on the pre-booking time, i.e., the time between booking h_i and pickup time t_i. The closer the pickup time approaches, the lower the fit requirements get because the potential of future requests decreases. Conversely, requests that arrive well in advance (up to 30 days) have higher fit requirements because the potential of incoming requests is still high. We define the time-dependent threshold through a step-wise function $\alpha(t_i - h_i)$, where the steps in the pre-booking times $t_i - h_i$ correspond to the shares of total requests. The function is calibrated using the irace framework applied to cross-validated training data (López-Ibáñez et al., 2016).

4.3. Machine learning application

After our ML models are trained on the displacement labels, we integrate the components into the dynamic process via the RFP framework. First, we describe the RFP algorithm and then introduce the feasibility check for the dynamic PM-DARP.

4.3.1. Request fit prediction algorithm

Algorithm 1 represents the application of the RFP. The input of our algorithm includes the current solution S and the incoming request r_i. When the request r_i enters the system, the algorithm conducts the following steps. The features (details in Section 4.2.2) based on the request information are generated. With these features the ML model predicts the request fit $\hat{\delta}_i$ of the request r_i (details in Section 4.2.3). If $\hat{\delta}_i$ exceeds the time-dependent dynamic threshold $\alpha(t_i - h_i)$ (details in Section 4.2.4), r_i is predicted to be a bad fit and rejected. Contrary, if $\hat{\delta}_i$ is below $\alpha(t_i - h_i)$, the feasibility of inserting the request into the current solution is checked (details in Section 4.3.2). Should the request r_i comply with all routing constraints, the solution $\hat{S}$ is updated and the request is inserted. Otherwise, if any constraint is violated, the request is rejected. The output is an updated solution $\hat{S}$.

Algorithm 1: Request fit predictor.

Input : Current solution S, request r_i
Output: Updated solution $\hat{S}$

```
features ← GenerateFeatures(r_i)                          // 4.2.2 Generate features
δ̂_i ← MachineLearningPredictor(features)                  // 4.2.3 Predict request fit
if δ̂_i < α(t_i − h_i) :                                   // 4.2.4 Dynamic threshold
    if FeasibilityCheck (S, r_i) :                        // 4.3.2 Feasibility check
        Ŝ ← Insert(S, r_i)
    else
        reject r_i
else
    reject r_i
return Ŝ
```

4.3.2. Feasibility check

Within the RFP framework, the feasibility of a request r_i is ensured by integrating a feasibility check that determines an efficient insertion into the existing solution S for the dynamic PM-DARP. The feasibility check can be performed using either a heuristic or an exact approach, as described below. In practice, the two approaches can also be combined sequentially: a fast heuristic is applied first, and the more computationally expensive exact solver is invoked only if the heuristic fails to identify a feasible solution.

The **heuristic approach** uses the cheapest insertion (CI) to provide a computationally efficient method for checking the feasibility of inserting a request r_i into the current solution S. Upon arrival of a request r_i, the algorithm evaluates all possible insertion positions for pickup and dropoff, ensuring that the pickup precedes the dropoff and that all constraints including time windows, vehicle capacity, and maximum ride time are satisfied (see Section 2.2). For each feasible insertion pair, it calculates the additional travel time introduced to the current solution S. Among these, the insertion that results in a minimal increase in travel time is selected. If such a feasible and cost-effective insertion exists, the solution is updated to $\hat{S}$ to include the request r_i; otherwise, the request r_i is rejected and the existing solution S remains unchanged. This greedy, local decision-making process ensures very fast runtimes and

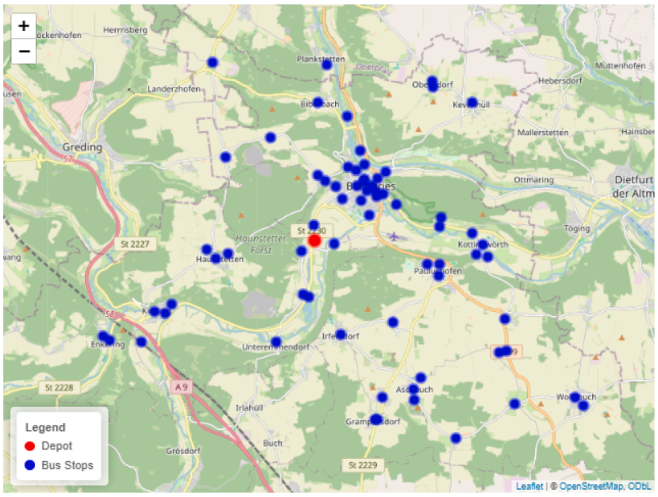


Fig. 3. Rural network structure.

is well suited for highly dynamic environments with frequent request arrivals. However, the approach is myopic in nature as it does not revise the placement of previously accepted requests, which may lead to suboptimal overall performance.

The **solver approach**, formulates the feasibility check as a compact event-based MIP (see Section 2.1) and applied as a sequential event-based solver (SeS). This formulation captures the insertion of a request r_i not by considering isolated insertion positions, but by re-optimizing the full solution S from the current system state. It allows the solution S to be fully restructured, i.e., adjusting the order, timing, and assignment of all currently accepted requests in order to incorporate the incoming request, provided that all routing constraints are met (see Section 2.1) and previously accepted requests remain feasible. The SeS returns an updated solution $\hat{S}$ that includes the request r_i in an optimal position if a feasible solution exists; otherwise, the request r_i is rejected and the solution S remains unchanged. Usually this method is computationally more demanding than the heuristic approach. However, for certain instance configurations, such as those with tight time windows or sparse request sets, the event-based graph remains compact, allowing the SeS to be solved efficiently.

5. Experimental design

This section introduces our real-world case study. Section 5.1 describes the PTP case and the experimental design. We then detail the data basis and explain the process of generating training, test, and validation data in Section 5.2. Finally, Section 5.3 introduces the benchmark approaches.

5.1. Case study

We have access to real passenger search data from a German PTP from June 1, 2022 to August 31, 2024. These contain requests that include information about the search date and time, pickup and drop-off locations, pickup and drop-off time, as well as the number of passengers. For each day, we use all search data that requested bookings for that specific day to plan a single tour. To account for the browsing behavior of customers, we filter the search data in two steps. First, we remove searches that resulted in an error code because the system configurations were violated (e.g., trips outside service hours or incorrect bus stops). Secondly, we remove searches from customers who looked for trips with the same pickup and drop-off districts during the same trip hour. Since a single request cannot participate in the same tour multiple times within an hour, these searches are removed because they are considered browsing behavior by potential passengers. To mimic the booking process, where requests are processed sequentially, we sort the data in ascending order of search date and time.

According to this process, we generate 26 months of valid searches that provide 122,289 requests with the potential to serve 146,332 passengers. This represents on average 154 valid requests per day and an average potential of 185 passengers per day. The area in which the search data are generated covers approximately 100 square kilometers and includes 70 fixed bus stops. This corresponds to a stop density of about 0.7 stops per square kilometer (approximately 1–1.2 km average spacing), which is consistent with transit planning guidelines for lower-density rural or peri-urban service areas and comparable to recent studies (Melo et al., 2024). The distance metrics for the total request population are as follows: Mean distance of 8.09 km with a standard deviation of 3.33 km. The minimum distance is 0.25 km while the maximum distance is 20.11 km. 25% of the requests cover distances less than 5.29 km and 75% cover distances less than 10.96 km. Fig. 3 shows the bus network and Fig. 4 presents the distribution of the distances requested.

The PTP operates on a limited budget having only two minibuses, each with eight seats. These minibuses operate for 18 h on weekdays, 14 h on Saturdays, and 12 h on Sundays. The minibuses ride in tours of 4 h, followed by a mandatory break of 0.5 h for the

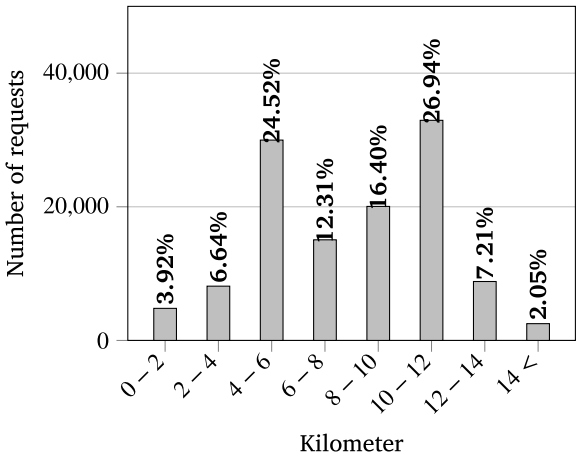


Fig. 4. Distance distribution.

driver after each tour. This four-hour tour duration defines the maximum tour length. Therefore, the minibuses execute four tours per day during the week and on Saturdays and only three tours on Sundays. The distances between the 70 bus stops are calculated using the haversine formula, multiplied by 1.3 to account for the road course. This value is in line with Merchan et al. (2020) and with the assumptions taken by the PTP. We assume that the minibuses travel at an average speed of 30 kph. The travel speed is considered to be deterministic because the minibuses operate in low traffic rural areas. To pickup or drop-off the passengers, the minibuses operate within a flexible time window of ±5 minutes. For instance, if a passenger wants to be served at 8 am, the minibus can pick up the passenger anytime between 7:55 am and 8:05 am. The same logic applies to the time window for the drop-off. The maximum passenger ride time is determined by the estimated travel time extended by 30% of the travel time. These assumptions are in line with the PTP to increase the flexibility of the system.

5.2. Test, training and validation data

The 26 months of valid searches are spread over 793 calendar days. To simulate real-world operations, we first train the ML models on 24 months of historical data and then apply the RFP on the two most recent (and previously unseen) months July and August 2024. As an operational day is divided into independent booking slots, the 62 test days represent 239 static PM-DARP instances, while the 731 training days correspond to 2818 static PM-DARP instances. Those training days are solved independently under PI to generate displacement labels (see Section 4.2.1). The instances contain on average 40 requests (median: 39, standard deviation: 17, interquartile range: 29–51), reflecting typical rural demand variability. For label generation, a runtime limit of 30 seconds per instance was imposed. Within this limit, 81.84% of the instances were solved to proven optimality. The remaining instances exhibit an average optimality gap of 15.0%, resulting in an overall average gap of 2.8% across all training instances. Although we cannot solve each instance to optimality, a small bias has only a marginal impact on downstream optimization performance, as shown in prior research (Gückel and Fontaine, 2025). To generate the PI solution for the test instances, the runtime limit was set to 15 minutes. Within this limit, we solved 95.7% of all instances to optimality. For the remaining 4.3%, optimality could not be proven; for these, the average optimality gap was 7.4%. On average, across all instances, the gap is approximately 0.33%.

Training and validation data: The 24 month of past data used to train the ML models represent 112,831 requests. These requests are labeled as described above. Moreover, we generate the features for all these requests as outlined in Section 4.2.2. To select the most appropriate model for the RFP framework, we allocate 80% of the training data (90,264 requests) for model training and use the remaining 20% (22,567 requests) to validate the prediction performance. To standardize the features, we apply z-score standardization to get normal distributed data ($\mu = 0$, $\sigma = 1$). For a more efficient hyperparameter tuning of the Random Forest and XGBoost models, we use tree-structured Parzen estimators, a Bayesian method that models the probability distribution of hyperparameters based on previous trials and adjusts them to improve results in future trials (Akiba et al., 2019). To identify stable architectures, we calibrate the hyperparameters with 5-fold cross-validation. Appendix C shows the final hyperparameters.

Test data: We evaluate our approach using data from July and August 2024, which together comprise 62 days. This period includes a representative mix of weekdays and weekends to ensure comprehensive coverage of operating conditions. The test days are reserved exclusively for the RFP application phase and are not used during model training. The benchmark methods are also evaluated on this test dataset.

5.3. Benchmark approaches

To evaluate the operational performance of the RFP framework, we compare our method against benchmarks on the 62 test days. We use three realistic and explainable approaches for the dynamic setting (CI, SeS, and a rule-based ablation RB-CI), and two

Table 1
Overview of benchmark approaches and their role in the analysis.

Benchmark	Feasibility mechanism	Role in the analysis
CI	Cheapest insertion (heuristic)	Realistic dynamic baseline (status quo)
RB-CI	Cheapest insertion (heuristic)	Rule-based ablation of RFP-CI
SeS	Sequential event-based solver (exact)	Realistic dynamic baseline
RFP-CI	Cheapest insertion (heuristic)	Isolates RFP effect under heuristic routing
RFP-SeS	Sequential event-based solver (exact)	Isolates RFP effect under exact routing
RFP*-SeS	Sequential event-based solver (exact)	Upper bound under perfect fit prediction
PI	Static solver, full information	Utopian upper bound

approaches (RFP*-SeS, PI) that, while unattainable in a dynamic environment, serve to quantify the maximum potential improvement. The framework is evaluated when combined with either CI (RFP-CI) or SeS (RFP-SeS) to quantify the effect of the request fit prediction.

Cheapest Insertion (CI): The CI serves as fast, heuristic-based benchmark. Due to its low computational complexity, it is well-suited for real-time applications. The CI is also embedded in the request fit predictor with cheapest insertion (RFP-CI) method. Therefore, we can isolate and quantify the effect of the RFP with a heuristic approach.

Rule-based cheapest insertion (RB-CI): The RB-CI extends the CI benchmark with a simple rule-based acceptance policy. Incoming requests are first evaluated according to a lexicographic rule that is identified in the feature importance analysis (see Section 6.1.2). Requests accepted by this rule are then routed using the CI heuristic. Because the rule is derived from the RFP itself, it is not an independent baseline but a simplified version of the RFP: it replaces the learned predictor and its time-dependent threshold with fixed rules and thus tests whether the learned predictor adds value beyond these rules.

Sequential event-based Solver (SeS): The SeS re-optimizes the entire solution solving the event-based MIP for each incoming request. While computationally more demanding, the SeS is expected to achieve higher solution quality. Since the SeS is also used in the request fit predictor with the sequential event-based solver (RFP-SeS), we can again directly evaluate the benefit of the RFP when combined with an exact insertion approach.

Perfect request fit predictor with sequential event-based solver (RFP*-SeS): The RFP*-SeS simulates an idealized version of the RFP-SeS approach, assuming perfect prediction based on a request's true impact on the overall solution quality. A request is classified as a good fit if its inclusion does not worsen the objective value of the optimal PI solution, i.e., only requests with $\delta_i = 0$ are considered for insertion. All other requests are excluded in advance. Requests still enter the system sequentially according to their original search times, preserving the online nature of the problem. The RFP*-SeS thus serves as an upper bound for the achievable performance when prediction is perfect.

Perfect information (PI): The PI solution serves as utopian benchmark to identify the potential of the search requests. This solution is created using the MIP. Assuming complete knowledge of all incoming requests, i.e., we solve the static problem. However, this solution cannot be achieved in reality because the incoming requests are not known in advance.

Table 1 summarizes the benchmark approaches, indicating for each its underlying feasibility mechanism, and its role in the subsequent analysis.

6. Results

We conduct several numerical experiments to analyze the performance of the RFP and its implications on the current business practice. In Section 6.1, we discuss the ML choice for the RFP and analyze the importance of its features. Section 6.2 presents our main results comparing the performance of RFP against benchmark approaches. In Section 6.3, we provide a managerial analysis and provide insights for PTPs. Appendices E and F, provide further information on robustness and scalability.

The RFP framework is implemented in Python, using pybind11 to integrate the CI in C++ for real-time decisions. The event-based MIP is solved with CPLEX (Version 22.1.1). The experiments are conducted on a system featuring an AMD Ryzen 5 5500U processor and 16 GB RAM.

6.1. ML model performance and interpretability

We compare the performance of the ML models with three simple benchmarks. Additionally, we assess feature importance to identify the most relevant features.

6.1.1. ML performance

The displacement label δ_i is a count variable which equals 0 when a request belongs to an optimal solution and takes on a non-negative integer value otherwise. As baseline, we include a naive model that always predicts zero and compare its performance to that of Poisson Regression and Negative Binomial Regression. Count data often display overdispersion, where the variance exceeds the mean, which renders Negative Binomial Regression a more appropriate choice than the standard Poisson Regression (McCullagh, 2019). Against these basic benchmarks, we compare the ML (Random Forest, XGBoost, and Neural Network). We evaluate the regression performance of the models using robust metrics such as Rooted Mean Square Error (RMSE) and Mean Absolute Error (MAE) (Géron, 2022).

Table 2
Machine learning performance.

Model	RMSE	MAE
Naive (always 0)	1.0447	**0.4931**
Poisson Regression	0.8796	0.6116
Negative Binomial	0.8810	0.6120
Random Forest	0.8552	0.5869
XGBoost	**0.8530**	0.5818
Neural Network	0.8725	0.5836

Table 3
XGBoost performance across different training data sizes.

XGBoost	2 years	1 year	6 months	3 months	1 month
RMSE	**0.8530**	0.8853	0.8730	0.9417	0.8772
MAE	**0.5818**	0.6107	0.6054	0.6320	0.6164

Table 2 lists the cross-validated prediction performance of the models. Among the classical statistical baselines, the Poisson and Negative Binomial regressions yield nearly identical predictive errors, with the Poisson formulation showing a marginal advantage in both metrics. Based on this predictive performance, a Poisson-based objective function (log-link) is adopted for the subsequent ML approaches. Therefore, XGBoost directly models count data, and the Neural Network uses an exponential output function for non-negative predictions. The tuned hyperparameters can be found in Appendix C.

XGBoost achieves the best overall performance, yielding the lowest RMSE and a competitive MAE. Although the naive model displays the lowest MAE, this metric is misleading, because the majority of the displacement values are zero, a zero-predicting baseline artificially minimizes absolute error while failing to capture any structural variation or true positive displacements in the data. While Random Forest and the Neural Network clearly outperform the baseline regressions, they do not match the predictive accuracy of XGBoost. The Neural Network's performance may be constrained by sample size, suggesting that the tabular nature of this displacement problem is better suited to tree-based ensemble methods.

To assess the impact of training data volume on predictive performance, we estimate the model using rolling historical windows ranging from one month to two years and evaluate accuracy on a fixed out-of-sample test set, July and August 2024. All windows end on 30 June 2024, immediately before the test period: two years (July 2022 to June 2024), one year (July 2023 to June 2024), six months (January to June 2024), three months (April to June 2024) and one month (June 2024). No observation from the test period is used for training or hyperparameter tuning. Table 3 reports RMSE and MAE across window lengths. Across both error metrics, the model trained on two years of historical data performs best, suggesting that longer observation periods improve predictive accuracy by capturing recurring seasonal and operational patterns, whereas shorter windows appear more sensitive to transient fluctuations.

Main finding: We select XGBoost for integration into the RFP with the complete two years of training data, as it demonstrates the best performance on cross-validated data.

6.1.2. Feature selection

This section analyzes the importance of features using the average Shapley values as proxy (Lundberg and Lee, 2017). Fig. 5 provides an overview of the importance of all 34 features. The most important feature in our model are the *passengers* (0.1769). The RFP clearly favors requests with more passengers, as serving a higher number directly contributes to maximizing the expected number of served passengers. In contrast, many of the time features have minimal independent impact because much of their information is implicitly captured by related features - for example, the categories *morning*, *noon*, and *evening* can be derived from *pickup_time*, and the slot distribution already encodes temporal variation. Interestingly, *slot_3* stands out among the time features, likely due to its association with the period of highest demand, resulting in a higher explanation value. While the nearest neighbor features, such as *pickup_nearest_neighbor* and *dropoff_nearest_neighbor*, tend to have less individual impact, the *gravity_distance* is highly significant. This feature reflects the centrality of a request relative to a designated service - a critical measure in rural areas where resources are more dispersed. Additionally, the feature representing the ten closest dropoff neighbors (*dropoff_10_nearest_neighbor*) plays an important role by offering insight into potential pooling opportunities. Aggregated demand features, which incorporate location via clusters, time, and demand information, also contribute to the model. However, their overall impact is modest.

We set a threshold of 0.02 on the SHAP feature importance values to filter out features with low explanation values (Lundberg and Lee, 2017). This threshold was chosen based on empirical evaluation and established practices in feature selection, ensuring that only those features with a meaningful contribution to the prediction of the request fit are retained. By excluding features that fall below this threshold, we narrow the model down to 16 key features that capture the most relevant aspects.

The dominance of the features *passengers* and *gravity_distance* provides a direct foundation for constructing a interpretable benchmark. We derive a rule-based acceptance policy that follows a lexicographic structure: requests with more than one passenger (average passengers per request: 1.29) are always accepted, whereas single-passenger requests are accepted only if their pickup location lies within a gravity-distance threshold. We evaluate alternative thresholds corresponding to the empirical gravity-distance quantiles Q60

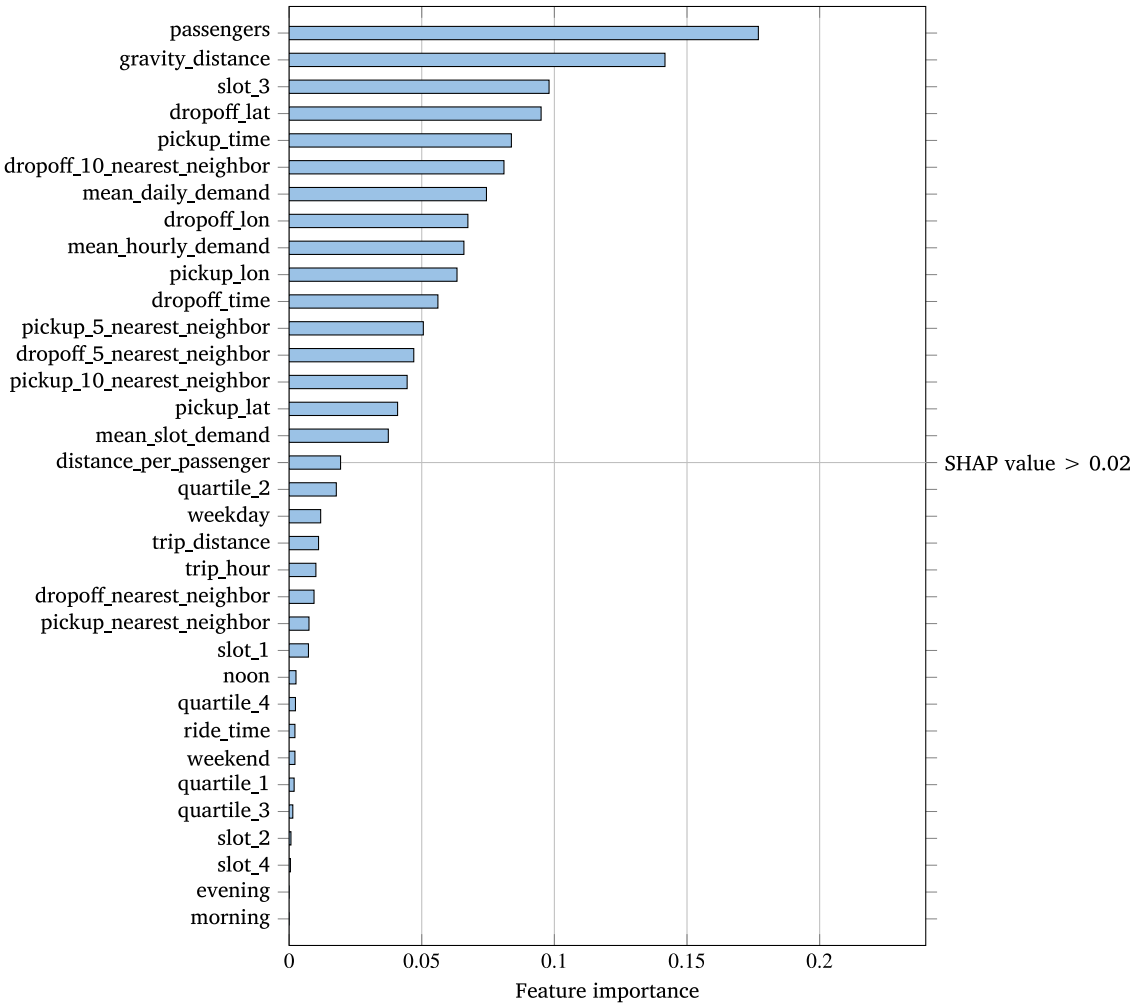


Fig. 5. Feature importance.

(< 6.3 km), Q90 (< 8 km), Q95 (< 9 km), and Q99 (< 12 km). Q95 yields the highest acceptance rate and is adopted in the approach alongside the CI for routing, yielding the RB-CI.

Main finding: The feature *passengers* is most critical, aligning with the objective, while *gravity_distance* captures the network structure essential for routing. Various time features contribute modestly, and although aggregated demand features provide useful context, they are less influential than capacity-related features.

6.2. Performance of RFP against benchmark approaches

We evaluate the operational performance of the RFP framework in a realistic application on the two test months July and August 2024. These test days consist of 239 booking slots with in total 9458 requests that have a potential of 11,529 passengers. The performance is assessed in comparison to five benchmarks, focusing on the expected number of served passengers, the computation times for real-time decisions, and the driven distance.

6.2.1. Expected number of served passengers

The main objective of the RFP is to maximize the expected number of passengers served in a dynamic setting. Fig. 6 presents the expected number of passengers served by the different benchmark approaches, both in absolute numbers per day and relative to the total available passengers.

Under the current business practice, the CI serves only 40.73% of available passengers, whereas the SeS improves this figure to 43.18%. Combined with the CI, the RFP serves exactly as many passengers as the SeS (43.18%). The learned predictor thus enables a fast heuristic to match an exact re-optimization approach, which is relevant for PTPs without access to commercial solvers or dedicated computing infrastructure. Relative to the pure CI, this corresponds to a gain of 2.45 pp (6.10%). Fixed acceptance rules

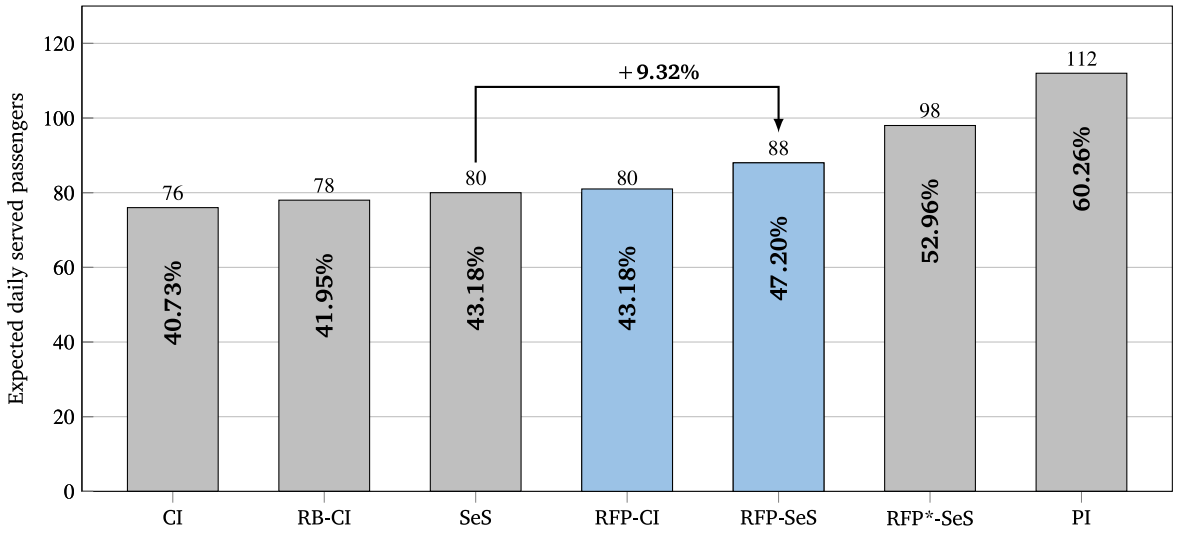


Fig. 6. Expected number of served passengers benchmark.

derived from the RFP (RB-CI) already reach half of this gain (41.95%, +1.22 pp), and the learned predictor adds the remaining 1.23 pp. Under cheapest-insertion routing, a simple rule thus captures a substantial part of the effect. The larger effect of the RFP arises in combination with the SeS: RFP-SeS serves 47.20% of the available passengers, 4.02 pp (9.32%) more than the pure SeS. Compared to the current business practice, this corresponds to 6.47 pp (15.98%) more passengers served.

Notably, these gains are not driven by accepting substantially more requests. While CI processes on average 60 requests per day, RFP-CI increases this number only marginally by one request per day. RFP-SeS also only adds one request per day to 66 accepted requests compared to SeS, but has a gain of 9.32% passengers. These comparisons show that performance differences are not primarily explained by the number of requests accepted, but by which requests are accepted. The RFP systematically prioritizes requests with higher passenger counts and greater compatibility with future routing decisions, allowing the solver to utilize scarce vehicle capacity more efficiently. Additional robustness analyses across different days and demand patterns are reported in Appendix E, showing that the RFP approach continues to produce performance gains even on weekends with shorter service hours and lower demand.

In a dynamic process, the benchmark RFP*-SeS (i.e., with perfect prediction) would serve 12.20% more passengers than RFP-SeS, quantifying the remaining potential if the request fit were determined flawlessly. Despite the rather high prediction uncertainty, both RFP approaches (RFP-CI and RFP-SeS) achieve superior results and outperform the realistic benchmarks (pure CI and SeS). The PI has a huge advantage over the other methods because it assumes that all requests are known. With this knowledge, 60.26% of the available passengers could be served, which is the absolute upper bound. The relatively modest performance gap between CI and PI indicates that the considered instances are structurally tight. This is primarily driven by rural characteristics such as tight pickup time windows, spatial dispersion of requests, and limited temporal overlap. Under these conditions, the potential for large-scale route restructuring is inherently limited, reducing the marginal benefit of full re-optimization. At the same time, the results highlight a structural capacity shortage in the system. Increasing the fleet size could potentially improve absolute service levels; however, rural public transport providers typically operate under strict budget constraints, making fleet expansion difficult in practice. Appendix F therefore examines scenarios with different fleet size (one and four vehicles). The results show that the proposed approach performs particularly well in capacity-constrained settings but also remains competitive as additional vehicles become available. Finally, we benchmark the learned threshold against an explicit look-ahead policy. A scenario-based one-step rollout policy (Goodson et al., 2013) does not exceed the myopic CI service level and remains below both RFP variants. Because demand is low and sporadic, sampled future requests frequently fail to materialize, so the policy anticipates displacement that does not occur. The acceptance threshold encodes a comparable anticipatory signal offline and is evaluated request-by-request against the currently committed state. Detailed results are reported in Appendix G.

Main finding: By applying the RFP-SeS, the PTP serves 15.98% more passengers than the status quo. Combined with the CI heuristic, the RFP matches the service level of the exact SeS, and fixed rules derived from the RFP already reach half of this gain. The larger effect of the RFP arises in combination with the SeS, where it serves 9.32% more passengers than the SeS alone.

6.2.2. Computation times

Potential passengers are actively waiting in the booking system and expect an immediate response for their search request. Therefore, computation times of the implemented approaches are crucial for practical applicability. Table 4 reports the average runtimes measured in milliseconds.

All benchmarks exhibit excellent computational efficiency, remaining well below one second per request and thus fully satisfying real-time operational requirements. As expected, the RB-CI provides the fastest response times, outperforming even the CI, as a subset of requests is directly rejected and therefore not evaluated in the routing procedure. Solver-based methods SeS are comparatively

Table 4
Computation time in milliseconds for the benchmarks.

Benchmark	CI	RB-CI	SeS	RFP-CI	RFP-SeS
Runtimes per request [ms]	2.48	1.90	82.47	14.84	35.74

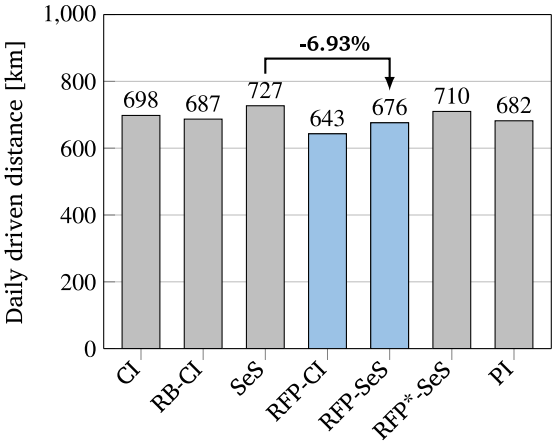


Fig. 7. Driven distance benchmark.

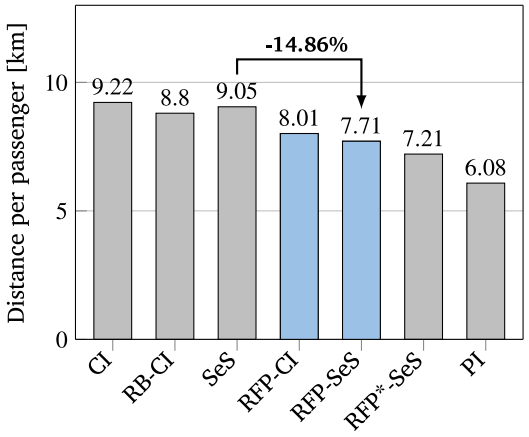


Fig. 8. Driven distance per passenger.

slower due to their higher computational complexity but still comfortably achieve runtimes suitable for real-time decision-making. Integrating ML-based predictions slightly increases computational effort: when combined with cheapest insertion (RFP-CI), it remains faster than the solver-based SeS-approaches but it is slower than the pure insertion method. When combined with the solver (RFP-SeS), runtimes remain substantially lower than for the pure solver-based approach, as the RFP filters out a subset of requests before the routing problem is evaluated. Additional scalability analyses with smaller and larger fleet sizes are reported in Appendix F. While computation times of solver-based approaches increase with fleet size due to the larger search space, runtimes remain well within real-time requirements.

6.2.3. Driven distance

The PTP operates within constrained resources and the limited number of minibuses can only cover limited distances during their service hours. Therefore, this section analyzes the effect of the benchmark approaches on the driven distance (in kilometers) on the test days. Fig. 7 presents the total driven distance on the 62 days.

Under the current business practice, the CI approach results in an average of 698 km driven per day over the test period. By incorporating the request-fit prediction, the RFP-CI variant not only increases the expected number of served passengers compared to CI but also reduces the daily distance by 7.87%. Similarly, the RFP-SeS method outperforms its pure benchmark (SeS) by serving more passengers and saving 6.93% in distance. These results further demonstrate the effectiveness of the request fit prediction in optimizing both service quality and distance coverage. Notably, both the idealized RFP*-SeS and the static PI yield comparable daily distances to our RFP approaches, indicating that the overall capacity in terms of driven kilometers is being used reasonably

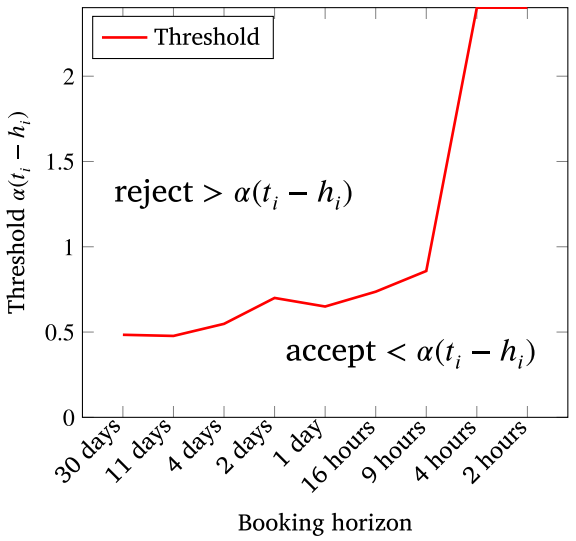


Fig. 9. Time-dependent threshold.

well. However, the RFP-CI variant does not fully exploit this capacity, suggesting improvement potential. The RB-CI reduces driven distance by systematically excluding spatially remote requests, lowering the average to 687 km (8.80 km per passenger). In contrast, the RFP achieves distance reductions while maintaining high acceptance by improving route composition rather than excluding distant requests outright.

Fig. 8 presents the driven distance per served passenger across the different benchmarks. Although CI and SeS remain viable dynamic approaches, they each require around 9 km per passenger, reflecting moderate efficiency. By contrast, our RFP framework reduces this metric substantially by 13.09% to the CI and by 14.86% to the SeS. At the same time, the figure highlights that the idealized RFP*-SeS and static PI achieve even shorter distances per passenger by selectively excluding longer trips. In practice, RFP-CI and RFP-SeS demonstrate a balanced approach as they do not systematically discriminate against outlying requests and still achieve notable distance reductions. This balance of efficiency and inclusivity can be vital in operation, where excluding distant passengers raises fairness concerns.

Main finding: By integrating the request fit prediction, RFP-CI reduces the distance per passenger by 13.09% compared to CI, while RFP-SeS achieves a 14.86% reduction relative to SeS. Notably, these gains come without systematically excluding longer trips.

6.3. Managerial analysis

In the managerial analysis, we analyze the applicability of the RFP from the PTP's uncertainty perspective to provide stable service levels throughout the entire booking horizon in Section 6.3.1. In Section 6.3.2, we analyze the fairness of the approach on passengers with longer distances.

6.3.1. Service level over the booking horizon

The 30-day booking horizon naturally favors early bookings, which can leave spontaneous passengers at a disadvantage. However, it would be beneficial to provide a stable service level over the booking horizon, effectively balancing early bookings with the needs of last-minute passengers. Therefore, we developed the time-dependent threshold. The service horizon is divided into nine intervals at 30 days, 11 days, 4 days, 2 days, 1 day, 16 hours, 9 hours, 4 hours, and 2 hours before departure. Each interval represents ~11% of the available bookings. The threshold $\alpha(t_i - h_i)$ is optimized with the training data according to the intervals using irace (López-Ibáñez et al., 2016). To isolate the contribution of this temporal structure from that of the fit prediction, Appendix D replaces the time-dependent threshold by static values under CI routing. No static threshold reaches the service level of the time-dependent one. A single value has to compromise between early bookings, for which much of the demand is still to come and a strict threshold is appropriate, and late bookings, which displace few future requests and can be accepted more leniently.

Fig. 9 illustrates how the threshold changes as the departure time approaches. At 30 days before departure, the threshold starts at 0.48 and increases to 0.70 two days prior to departure. The threshold temporarily decreases to 0.65 one day before departure, indicating that the training data contains a comparatively higher share of well-fitting requests in this interval, allowing irace to calibrate a stricter, more selective threshold without sacrificing service level. Until nine hours before the departure, the threshold slightly increases to 0.86. As the potential of future passengers declines, the threshold sharply increases four hours before the departure to 2.40 and remains rather stable afterwards. This means the RFP becomes irrelevant nearing the departure time because almost no request should be rejected before checking for feasibility. The positive effect of the time-dependent dynamic threshold can be observed in Fig. 10 where the RFP-SeS is compared against the SeS and the RFP*-SeS benchmark over the booking horizon. Here, the service

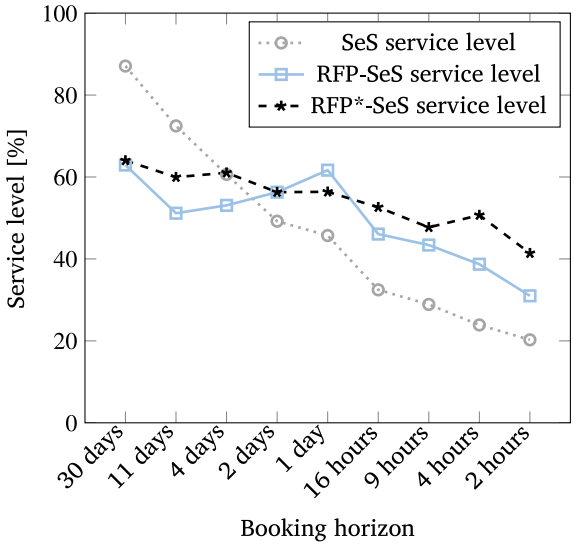


Fig. 10. Service level booking horizon.

level at each interval is defined as the share of all passengers with a pre-booking time falling into that specific interval who are ultimately served, rather than a cumulative measure over the entire horizon; each of the nine intervals is evaluated independently against the passengers available within it.

The SeS starts at an impressive 87.08% at 30 days before departure but then deteriorates sharply to just 20.26% in the final two-hour window, highlighting its declining capacity as the departure time nears. While the SeS aggressively captures early bookings, it fails to reserve capacity for spontaneous, later-stage demand. The proposed RFP-SeS approach adopts a more cautious strategy early on, preserving capacity for potential incoming requests. It begins at 62.92% at 30 days and remains at 53.08% four days before. Notably, its performance improves significantly in the critical booking window: reaching 56.30% at 2 days, peaking at 61.67% at 1 day before. Even 2 hours before departure, RFP-SeS serves 31.00% of the passengers - substantially outperforming the 20.26% achieved by SeS. When compared to the idealized RFP*-SeS with the perfect prediction, our approach tracks remarkably closely and even outperforms it one day before departure. In the critical final hours, RFP*-SeS serves 47.71% at nine hours, 50.69% at four, and 41.39% at two hours out, demonstrating potential for further gains through improved fit prediction.

Main finding: Over the entire booking horizon, RFP-SeS successfully maintains a balanced service level by preserving capacity during the early stages and relaxing fit requirements as departure nears. This strategy not only achieves performance comparable to the idealized RFP*-SeS but also significantly outperforms traditional benchmarks with first-come-first-served strategies in the critical final hours for spontaneous trips.

6.3.2. Passenger fairness

Although the RFP maximizes the expected number of served passengers, all tests show reduced driven distance as a side effect. We therefore examine two questions: whether acceptance depends on trip distance (distance equity), and whether any bus stop or origin–destination pair is excluded from service (coverage completeness). The RFP rejects a total of 23.01% of the requests that represent 19.21% of the potential passengers. However, of the 2215 rejected passengers 50.83% are also rejected by the SeS without the ML component. Therefore, the true rejection rate would be 9.45% of all passengers. For the fairness evaluation, we focus on the comparison between the SeS and the RFP-SeS. The RFP predicts the request fit based on geographical features, such as the gravity distance. Therefore, the suspicion is obvious that the approach prefers shorter distances over longer trips. Fig. 11 displays the expected served passengers, infeasible or directly rejected by the RFP compared to the served or rejected requests (because of infeasibility) of the SeS. The requests are grouped by their distance within 2 km intervals.

The SeS does not take into consideration request information and inserts requests according to the pre-booking time on a first-come, first-served basis. Nevertheless, the SeS approach shows an acceptance rate that decreases with increasing distance, ranging from 53.32% in the 0–2 km interval down to 31.16% for trips over 14 km. As the SeS can still alternate the positions of the requests in the solution, shorter distances fit better because they consume less capacity. In contrast, since the PI favors shorter distances (see Section 6.2.3), the RFP-SeS model trained on PI data also tends to prefer shorter requests. Our RFP-SeS achieves its highest acceptance rate of 61.31% in the 2–4 km interval and remains relatively robust up to 12 km (between 42.08% and 55.57%), though it decreases to 18.83% at 12–14 km and 22.10% for distances over 14 km. Relative to SeS, RFP-SeS accepts a higher share of requests in every interval up to 12 km except 8–10 km, but a clearly lower share beyond 12 km (18.83% vs. 39.61% at 12–14 km; 22.10% vs. 31.16% above 14 km). With respect to distance equity, the RFP is therefore not neutral: it shifts acceptance toward short- and medium-distance requests. From a strategic perspective, RFP-SeS accepts requests based on how well they fit into existing capacity and routes, thus providing flexibility for short- and medium-distance trips while still allowing for longer trips at a lower acceptance

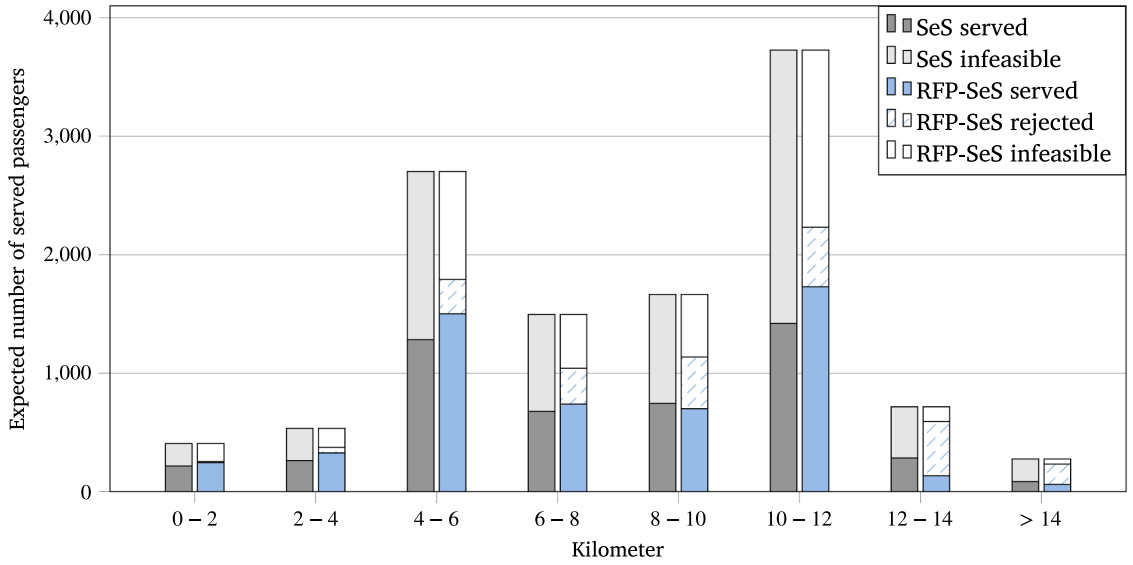


Fig. 11. Fairness of the RFP-SeS vs. SeS on distance-clustered number of served passengers.

rate. With respect to coverage completeness, the picture differs. Neither approach fully excludes any specific bus stops or distances in all cases. Consequently, the bus stop design remains unaffected, since no origin–destination pair is universally rejected. However, RFP-SeS leverages its acceptance logic more effectively at shorter distances and does not rely solely on a first-come, first-served basis. In contrast, a spatial filtering policy such as the RB-CI would have substantial implications for network coverage. Applying a gravity-distance threshold at the Q95 level (< 9 km) would exclude 14 bus stops, corresponding to approximately 20% of all stops in the network. These stops are located in sparsely populated peripheral areas, suggesting that such a policy would effectively deny mobility access to a non-negligible share of rural residents with single-passenger requests. This highlights that distance-based exclusion strategies improve efficiency primarily by shrinking the service area, whereas the RFP framework improves efficiency by better utilizing available capacity while preserving mobility access across the entire network.

Main finding: RFP-SeS does not provide distance equity: beyond 12 km it accepts requests at a lower rate than SeS (18.83% vs. 39.61% at 12–14 km; 22.10% vs. 31.16% above 14 km). It does, however, preserve coverage completeness: no bus stop and no origin–destination pair is excluded from service, whereas a gravity-distance threshold at the Q95 level would remove 14 stops, about 20% of the network. The efficiency gain of the RFP thus results from better use of capacity rather than from a smaller service area.

7. Conclusion

In this paper, we study the problem of predicting the request fit in dynamic PM-DARPs, a challenge that necessitates real-time decision-making because customers expect immediate responses in the booking system. We introduced the RFP framework, which integrates supervised ML models with classical optimization methods. By employing a displacement label based on problem-specific features and a time-dependent threshold, our framework effectively captures the dynamic nature of the problem. The modular design of the RFP enables seamless integration with either an exact solver or a heuristic method, ensuring solution feasibility and robust performance in real-time applications. Tested on 26 months of real data from a German PTP, we find that the RFP improves outcomes primarily by changing which requests are accepted rather than how many, prioritizing higher-occupancy, better-fitting requests to serve 15.98% more passengers than the status quo and 9.32% more than a sequential solver benchmark, while also reducing driven distance by 6.93%. Beyond this efficiency gain, the time-dependent threshold sustains a stable service level across the entire booking horizon, substantially outperforming a first-come-first-served solver and thereby preserving capacity for spontaneous ride requests. We further show that these gains do not come at a fairness cost: the RFP does not exclude any bus stop from service, unlike a comparable distance-threshold policy, and roughly half of its apparent rejections would occur regardless of the learned prediction, due to feasibility constraints alone.

The prediction of the passenger fit provides the PTP with an approach that is tailored to the challenges of rural areas. Several exciting business model features could be explored. Based on the request fit, the PTP could introduce a dynamic pricing scheme that offers incentives for beneficial routes and times to maximize the coverage of their service. Moreover, the PTP could use knowledge of well-fitting requests from one area when launching new on-demand minibus services in similar rural settings. An important limitation concerns the scope of observable demand in rural areas. Our analysis relies on data from an existing PTP, which captures realized and searched requests but not potential passengers who do not consider on-demand services, a particularly relevant issue in car dependent rural regions. Although search requests broaden the observed demand base, unobserved latent demand may still bias the learned request fit patterns. Future research could incorporate simulated or synthetic demand based on population, travel survey, or

car usage statistics to examine how activating latent demand and denser request networks affect rejection and acceptance dynamics. Finally, we note that our analysis assumes a homogeneous fleet, which reflects the operational reality of the studied system. Extending the framework to heterogeneous fleets would require accounting for vehicle-specific characteristics in both the static optimization used for label generation and the subsequent routing and feasibility checks (e.g., vehicle assignment, capacity, or operating costs). The learning-based decision logic itself would remain unchanged, provided that the model is trained on data reflecting the fleet configuration used during deployment.

Academics can build on our approach of solving an offline optimization problem with PI and training supervised ML models for real-time decisions. This learning component can be embedded in dynamic, time-critical processes and applied to other routing-based problems. Our modular framework supports both a MIP solver for optimal solutions and heuristics when solvers are unavailable, enabling performance comparisons between exact and approximate strategies. Future work could expand feature sets to improve prediction accuracy and better handle request uncertainty, thereby increasing the number of served passengers. While reduced distance was a positive by-product, focusing on environmental objectives could yield further insights for PTPs in promoting sustainable rural mobility. The objective function could directly include an explicit equity term, alongside served passengers and routing cost, potentially yielding a Pareto frontier between efficiency and equity that a PTP could select a point on according to its policy priorities.

CRediT authorship contribution statement

Simon Mader: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization; **Pirmin Fontaine:** Writing – review & editing, Validation, Supervision, Resources, Project administration, Methodology, Investigation, Funding acquisition, Conceptualization; **Stefan Voigt:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Methodology, Investigation, Formal analysis, Data curation, Conceptualization.

Funding

The research of the first author is supported by the German Federal Ministry for Digital and Transport (BMDV) within the funding project newMIND for the mobility transformation in Ingolstadt, Bavaria.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors acknowledge that this study is financially supported by the project ”VGI newMIND”, funded by the German Federal Ministry of Digital and Transport (BMDV) to promote the public transport association in Ingolstadt (VGI). The authors thank the VGI project team of the on-demand bus services for sharing their data, as well as the editor and the anonymous reviewers, whose constructive comments helped to significantly improve the paper.

Data availability

The authors do not have permission to share data.

Appendix A. Nomenclature

Table A.1 summarizes the notation used throughout the static and dynamic PM-DARP formulation in Section 2, as well as the request fit predictor components introduced in Section 4.

Table A.1
Notation used in the PM-DARP formulation and RFP framework.

Category	Symbol	Description
Sets & indices	$R = \{r_1, \dots, r_n\}$	Set of requests
	K	Set of available vehicles
	k	Generic index referring to a vehicle in K
	$J = P \cup D \cup \{0^+, 0^-\}$	Set of all locations
	$P = \{1^+, \dots, n^+\}$	Set of pickup locations
	$D = \{1^-, \dots, n^-\}$	Set of drop-off locations
	$\{0^+, 0^-\}$	Depot locations (start / end of vehicle route)
	V	Set of events in the event-based graph
	A	Set of arcs in the event-based graph
	$\lambda^{\text{in}}(v),\ \lambda^{\text{out}}(v)$	Set of incoming / outgoing arcs of event v
Request parameters	$i^+,\ i^-$	Pickup / drop-off location of request i
	q_i	Number of passengers of request i
	t_i	Preferred pickup time of request i
	h_i	Booking (search) time of request i
	s_i	Service time required at pickup and drop-off of request i
	$[e_j, l_j]$	Time window during which service must begin at location j
	t_{i^+,i^-}	Direct travel time from pickup to drop-off of request i
	ϵ	Maximum additional ride time factor
	L_i	Maximum allowed ride time for request i
Parameters	c_a	Distance associated with arc $a \in A$
	γ	Scalar cost factor weighing routing distance in the objective
	H	Booking horizon
	Q	Maximum number of requests on board (event dimensionality)
	$(v_1, \dots, v_Q)$	Q-tuple representing an event
	t_{ij}	Travel time between locations $i, j \in J$
	M_{ij}	Sufficiently large constant
Decision variables	$y_i \in \{0, 1\}$	Acceptance decision for request i
	$x_a \in \{0, 1\}$	Arc-usage decision for arc $a \in A$
	$B_j \geq 0$	Start-of-service time at location j
	$S,\ \hat{S}$	Current / updated solution in the dynamic problem
RFP framework	F^{PI*}	Optimal objective value of the static PM-DARP
	F_i^{PI}	Objective value when request i is forced into the solution
	δ_i	Displacement label of request i
	$\hat{\delta}_i$	Predicted request fit (displacement) of request i
	$f(r_i)$	Learned prediction function mapping request i to $\hat{\delta}_i$
	$\alpha(t_i - h_i)$	Time-dependent threshold as function of pre-booking time
	$\pi,\ \pi^*$	(optimal) policy in the dynamic problem (MDP formulation)

Appendix B. Features

The features are introduced in Section 4.2.2 and the total feature set is listed in Table B.1. For the feature importance, we refer the reader to Fig. 5 in Section 6.1.

Table B.1
Geographic, time, capacity, and aggregated demand features.

#	feature	description
1	pickup_lat	pickup location latitude
2	pickup_lon	pickup location longitude
3	dropoff_lat	drop-off location latitude
4	dropoff_lon	drop-off location longitude
5	trip_distance	booked distance from pickup to drop-off location
6	gravity_distance	distance from the network center to the pickup location
7	pickup_nearest_neighbor	distance from the pickup location to the nearest bus stop
8	pickup_5_nearest_neighbor	average distance from the pickup location to the five nearest bus stops
9	pickup_10_nearest_neighbor	average distance from the pickup location to the ten nearest bus stops
10	dropoff_nearest_neighbor	distance from the drop-off location to the nearest bus stop
11	dropoff_5_nearest_neighbor	average distance from the drop-off location to the five nearest bus stops
12	dropoff_10_nearest_neighbor	average distance from the drop-off location to the ten nearest bus stops
13	pickup_time	booked pickup time in minutes
14	dropoff_time	booked drop-off time in minutes
15	ride_time	estimated ride time from pickup time to drop-off time
16	trip_hour	pickup time rounded to the full hour
17	morning	request with pickup times before noon
18	noon	request with pickup times between noon and 5pm
19	evening	request with pickup times after 5pm
20	weekday	request on Monday, Tuesday, Wednesday, Thursday, or Friday
21	weekend	request on Saturday or Sunday
22	slot_1	request within four hours after the start of the first tour of the day
23	slot_2	request within four hours of the second tour, starting 30 minutes after the first tour has ended
24	slot_3	request within four hours of the third tour, starting 30 minutes after the second tour has ended
25	slot_4	request within four hours of the fourth tour, starting 30 minutes after the third tour has ended
26	quartile_1	the first hour of each tour (i.e., slot_1, slot_2, slot_3, slot_4)
27	quartile_2	the second hour of each tour (i.e., slot_1, slot_2, slot_3, slot_4)
28	quartile_3	the third hour of each tour (i.e., slot_1, slot_2, slot_3, slot_4)
29	quartile_4	the fourth hour of each tour (i.e., slot_1, slot_2, slot_3, slot_4)
30	passengers	number of passengers on the request
31	distance_per_passenger	booked distance (feature #5) divided by the number of passengers on the request (feature #30)
32	mean_hour_demand	mean hourly demand per cluster per day (Monday to Sunday) of the training data
33	mean_slot_demand	mean slot (four hours) demand per cluster per day (Monday to Sunday) of the training data
34	mean_day_demand	mean daily demand per cluster per day (Monday to Sunday) of the training data

Appendix C. Hyperparameters

The hyperparameters of the ML models were tuned during the training phase. In Tables C.1–C.3 the tuned hyperparameters are listed. The feature scaling and 5-fold cross validation was conducted with the scikit-learn package (Pedregosa et al., 2011). For the hyperparameter tuning of the Random Forest and the XGBoost, we used the optuna package (Akiba et al., 2019).

The Random Forest was implemented with skicit-learn (Pedregosa et al., 2011) and resulted in the hyperparameters in Table C.1.

Table C.1
Random Forest hyperparameters.

hyperparameter	values
n_estimators	704
max_depth	17
min_samples_leaf	5
min_samples_split	11
max_features	log2

The XGBoost model was implemented with the original dmlc XGBoost package (Chen and Guestrin, 2016) and resulted in the following hyperparameters in Table C.2.

The Neural Network was constructed using the TensorFlow Keras API (Chollet, 2015) according to best practice architectures in Table C.3.

Table C.2
XGBoost hyperparameters.

hyperparameter	values
objective	count:poisson
eval_metric	poisson-nlogik
learning_rate	0.01
max_depth	6
min_child_weight	2
colsample_bytree	0.8
subsample	0.8
lambda	1.0
alpha	0

Table C.3
Neural Network hyperparameters.

hyperparameter	values
input layer	units = 256, activation = 'relu'
hidden layer 1	units = 256, BatchNormalization, Dropout(0.2), activation = 'relu'
hidden layer 2	units = 128, BatchNormalization, Dropout(0.2), activation = 'relu'
hidden layer 3	units = 64, BatchNormalization, Dropout(0.2), activation = 'relu'
hidden layer 4	units = 32, BatchNormalization, Dropout(0.2), activation = 'relu'
output layer	units = 1, activation = 'exponential'
optimization	Adam(learning_rate = 0.0001)
loss function	Poisson()
metrics	RootMeanSquaredError()

Appendix D. Static versus time-dependent threshold

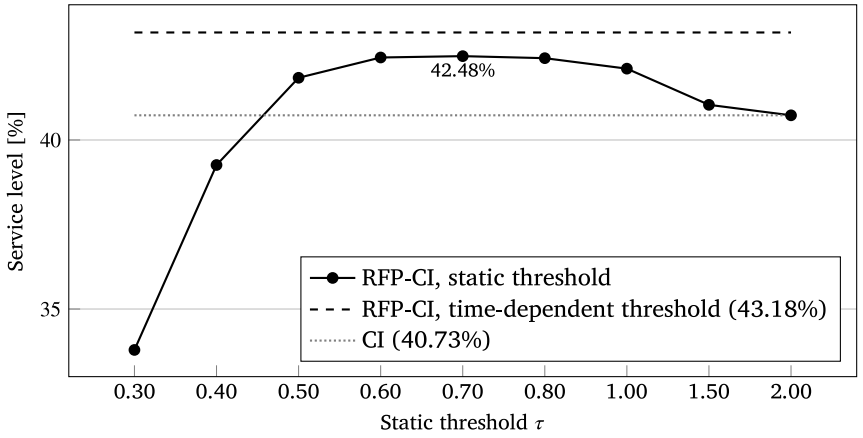


Fig. D.1. Service level of RFP-CI with a static threshold τ on the predicted displacement (solid line), compared with the time-dependent threshold (dashed line) and cheapest insertion without rejection (dotted gray line); 2 vehicles, 62 test days.

The RFP combines two adaptive mechanisms: the predicted displacement of a request and a threshold that depends on the time between booking and pickup (Section 4.2.4). To separate their contributions, we replace the time-dependent threshold by a static threshold τ and keep the prediction model and the cheapest-insertion routing unchanged. A request is rejected if its predicted displacement is at least τ, regardless of when it is booked. In the time-dependent threshold, requests booked less than one hour before

pickup are never rejected. We therefore evaluate each static threshold with and without this rule. All variants are evaluated on the 62 test days with 2 vehicles.

Fig. D.1 reports the results. The service level follows an inverted U-shape in τ. Strict thresholds reject too many requests and fall below CI ($\tau \leq 0.4$), while lenient thresholds approach CI because almost no request is rejected. Between $\tau = 0.4$ and $\tau = 0.8$, the service level is nearly constant at about 42.48%. Since it is selected on the test period, this value overstates what a static threshold would reach in deployment. The time-dependent threshold reaches 43.18%. Of its gain of 2.45 pp over CI, the fit prediction with the best static threshold accounts for 1.75 pp and the temporal structure of the threshold for at least 0.70 pp. Taken together, roughly two thirds of the gain of RFP-CI over CI can be attributed to the fit prediction and one third to the temporal structure of the threshold. This attribution is approximate, since the prediction can only act through a threshold, the time-dependent values are calibrated to the same prediction model, and the static reference threshold is selected on the test period, but it shows that both mechanisms contribute to the result.

Appendix E. Robustness

This section examines the robustness of the results across different days of the week. Table E.1 reports the performance of RFP-SeS compared to SeS in the standard case with two vehicles, distinguishing between weekdays, Saturdays, and Sundays. The table presents the expected number of served passengers and driven distance for both approaches as well as the relative differences (Δ).

Table E.1
Performance of RFP-SeS vs. SeS per day.

Instances		SeS		RFP-SeS		RFP-SeS vs. SeS	
Day	Available passengers	Served passengers	Driven kilometer	Served passengers	Driven kilometer	Δ passengers	Δ kilometer
Weekday	199	83	777	91	722	9.69%	-7.07%
Saturday	187	85	640	92	584	9.20%	-8.65%
Sunday	112	62	540	66	521	6.71%	-3.54%
Average	186	80	727	88	676	9.32%	-6.93%

The performance improvements of RFP-SeS are consistent across all day types. On weekdays, the approach serves 9.69% more passengers while reducing driven distance by 7.07%. On Saturdays, the improvements amount to 9.20% more served passengers and 8.65% lower driven distance. On Sundays, where service hours are shorter and demand is lower, the approach still achieves gains of 6.71% in served passengers and a reduction of 3.54% in driven distance. Overall, the results indicate that the performance improvements remain stable across different demand patterns.

Figs. E.1 and E.2 illustrate the distribution of performance differences across individual days. For served passengers, the majority of instances show positive improvements of RFP-SeS over SeS, with only a small number of slightly negative outliers. The quartile ranges confirm that improvements are consistently positive across weekdays, Saturdays, and Sundays. A similar pattern can be observed for driven distance. Across all day types, the median differences indicate a reduction in distance when using RFP-SeS. The reductions are most pronounced on weekdays and Saturdays, while Sundays show slightly smaller improvements. Overall, these results suggest that the performance improvements of RFP-SeS over SeS are robust across different day types and demand conditions.

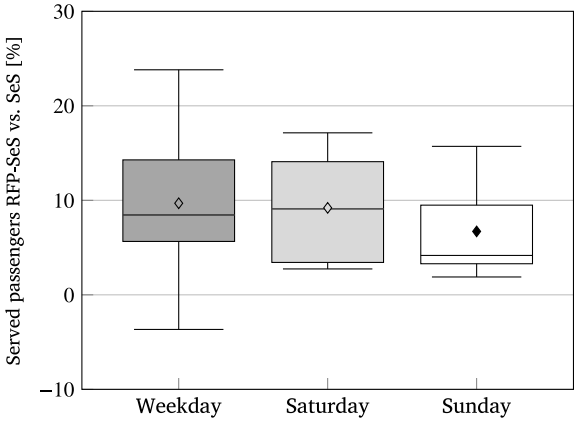


Fig. E.1. Served passengers.

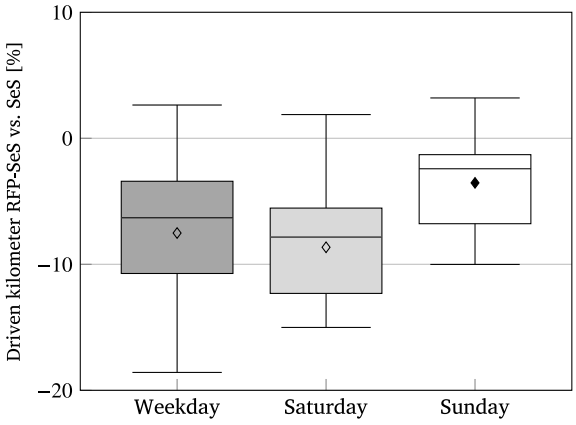


Fig. E.2. Driven distance.

Appendix F. Scalability

To assess the scalability of the proposed approach, we evaluate fleets of varying sizes. Specifically, we consider fleets of one, two, and four vehicles while keeping the trained request fit predictor unchanged. Table F.1 reports the results for the CI, RFP-CI, SeS, RFP-SeS, and PI benchmarks. This experiment evaluates whether the learned request fit remains effective as system capacity increases or decreases relative to demand, without retraining the model.

Table F.1
Scalability analysis of different benchmarks for varying numbers of vehicles.

# Vehicles	Benchmark	Served passengers (% service level)	Computation times [ms]	Distance per passenger [km]
1	CI	41 (22%)	1.61	8.93
	RFP-CI	47 (25%)	13.60	7.46
	SeS	41 (22%)	35.10	9.03
	RFP-SeS	48 (26%)	19.35	6.50
	PI	68 (37%)	–	5.15
2	CI	76 (41%)	2.48	9.22
	RFP-CI	80 (43%)	14.84	8.01
	SeS	80 (43%)	82.47	9.05
	RFP-SeS	88 (47%)	35.74	7.71
	PI	112 (60%)	–	6.08
4	CI	130 (70%)	2.27	9.40
	RFP-CI	129 (69%)	14.37	9.18
	SeS	143 (77%)	133.33	8.96
	RFP-SeS	146 (79%)	131.85	8.50
	PI	162 (88%)	–	7.37

Across all fleet sizes, the relative ranking of the benchmarks remains largely stable. In particular, the RFP-based approaches outperform their non-learning counterparts in terms of served passengers while maintaining competitive distance efficiency. Distance per passenger is generally reduced under the RFP-based approaches, particularly in the constrained settings with one or two vehicles, indicating that the improved acceptance decisions do not come at the expense of routing efficiency. The improvements are most pronounced in resource-constrained settings with one or two vehicles, where acceptance decisions are particularly critical and inefficient capacity allocation can significantly reduce the number of served passengers. In contrast, as fleet size increases and capacity becomes less scarce, the performance differences between the approaches gradually narrow. Consistent with this observation, the gap to the offline PI benchmark also decreases with larger fleets, reflecting that additional vehicles relax routing constraints and reduce the impact of selective request acceptance.

The models are not retrained for the different fleet sizes. For the four-vehicle setting, we slightly relax the acceptance threshold, as the threshold learned in the baseline configuration (two vehicles) becomes overly restrictive in higher-capacity systems. Interestingly, while RFP-CI shows slightly weaker performance when capacity is abundant, RFP-SeS continues to achieve small but consistent improvements. This suggests that the proposed approach is particularly beneficial in congested systems with scarce resources, while still providing modest gains when capacity constraints are less binding.

Computation times increase with fleet size but remain well below one second per request, confirming the suitability of the proposed approaches for real-time booking environments.

Appendix G. Scenario-based rollout benchmark

To assess whether the learned acceptance threshold approximates an explicit look-ahead, we implement a scenario-based one-step rollout policy (Goodson et al., 2013) as an additional benchmark. For every incoming request r_i, the policy proceeds as follows. First, it checks the feasibility of inserting r_i into the currently committed routes via cheapest insertion. Second, it samples S future request streams for the remainder of the tour block from an empirical demand model estimated on the training period only. Third, for each sampled scenario it rolls out the cheapest-insertion base policy under the two branches accept r_i now and reject r_i now over the same future realization, evaluating the objective (served passengers $-\gamma\cdot$ driven km). Fourth, it accepts r_i if the expected marginal objective is non-negative, that is, if the expected displacement of future passengers does not exceed the contribution of r_i itself. The base policy is CI, identical to that underlying the CI and RFP-CI benchmarks, so that differences in performance are attributable to the anticipatory mechanism rather than to the insertion heuristic.

Table G.1
Performance of the scenario-based rollout policy with 2 vehicles.

Policy	Served passengers	Computation time [ms]	Distance per passenger [km]
Look-ahead ($S = 5$)	73 (39%)	316	7.47
Look-ahead ($S = 10$)	74 (40%)	563	7.26
Look-ahead ($S = 15$)	75 (41%)	1451	7.06
CI	76 (41%)	2.48	9.22
RFP-CI	80 (43%)	14.84	8.01
SeS	80 (43%)	82.47	9.05
RFP-SeS	88 (47%)	35.74	7.71

Table G.1 reports the results. Evaluated out-of-sample, the rollout policy does not exceed the myopic cheapest-insertion service level: it serves 73–75 passengers across $S \in \{5, 10, 15\}$ against 76 for CI, approaching the myopic level from below as the scenario count grows, and remains clearly below both RFP variants (RFP-CI: 80; RFP-SeS: 88). It nonetheless requires 316–1451 ms per decision, against 15–36 ms for the RFP variants and approximately 2.5 ms for CI. The lowest distance per passenger of the rollout (7.06–7.47 km) does not indicate superior routing efficiency: by rejecting requests with high expected displacement cost, the policy selectively accepts well-fitting requests, which mechanically lowers per-passenger distance while serving fewer passengers. RFP-SeS attains a comparable 7.71 km while serving 17% more passengers.

As a diagnostic, we re-estimate the demand model on the evaluation period, granting the policy information unavailable in deployment; the service level then rises to approximately 44% at $S = 30$, above the myopic baseline but still below RFP-SeS. The out-of-sample shortfall is thus attributable to the difficulty of forecasting sparse demand rather than to an implementation error.

Because rural demand is low and sporadic, sampled future requests frequently fail to materialize, so the rollout anticipates displacement that does not occur and the additional online computation yields no additional served passengers. The displacement label encodes a comparable anticipatory signal at a lower online cost. Two qualifications follow. First, the threshold is not a rigid routing plan: it is evaluated request-by-request against the currently committed state and therefore remains adaptive online. Second, this does not establish general dominance over look-ahead policies: under denser, more predictable demand, where sampled scenarios are more informative, an explicit rollout may close or reverse the gap.

Appendix H. Reinforcement learning experiments

Section 3.4 argues that experience-based policies are poorly suited to the PM-DARP because coupled pickup and drop-off time windows make the admissible action set narrow and state-dependent. We test this claim with a policy-gradient agent PPO (Schulman et al., 2017) and a value-based agent DQN, the method used by Chen et al. (2022).

We model the accept/reject control as a Markov decision process (Ulmer et al., 2020) in which one episode corresponds to a tour block and decision epochs follow the arrival order of requests. The 40-dimensional state comprises the 26 request features of the RFP, including the pre-booking time; a summary of each vehicle's committed route (assigned requests and passengers, planned distance and travel time, latest scheduled event); the CI feasibility and insertion cost of the current request; and the progress within the block together with an estimate of the remaining demand. The progress uses the realized block size, which favors the agent. An accepted feasible request is inserted via CI and yields its number of passengers minus γ times the incremental distance; accepting an infeasible request has no effect, so no reward shaping is needed. The agent thus observes strictly more information than the RFP threshold and optimizes the objective that the displacement label encodes.

Both agents are trained on the same data as the RFP: all 2,913 tour blocks of the historical period outside the test window, sampled uniformly with replacement. Each run comprises 100,000 decisions, corresponding to about 1,290 episodes. For PPO, we train one policy for each entropy coefficient in $\{0.0, 0.01, 0.05, 0.1, 0.2\}$ to rule out under-exploration. For DQN, we train three seeds. Exploration follows an ε-greedy schedule with ε decreasing from 1.0 to 0.05 over the first 20% of training. In this phase, the agent rejects about half of all feasible requests at random, so the effect of rejections is observed regardless of the learned policy. Table H.1 lists the remaining settings. Each policy is evaluated deterministically on the 239 held-out tour blocks of July and August 2024 used for all other benchmarks. To exclude differences in implementation, we compare the agents with CI evaluated in the same environment.

Table H.1
Training settings of the reinforcement learning agents.

	PPO	DQN
Training blocks	2,913	
Decisions / episodes per run	100,000 / approx. 1,290	
Networks	MLP, 2×128 units	
Discount factor	0.997	
Learning rate	$3 \cdot 10^{-4}$	10^{-4}
Exploration	entropy coefficient $\{0, 0.01, 0.05, 0.1, 0.2\}$	ε: 1.0 → 0.05 over 20% of training
Further settings	GAE $\lambda = 0.95$, clip range 0.2, rollout 2048, minibatch 256, 10 epochs	replay buffer 100,000, minibatch 256, target update every 5,000 decisions
Runs	one per entropy coefficient	three seeds
Implementation	Stable-Baselines3	

The PPO agent accepts slightly more passengers than the CI and reaches the service level of 41.50%. Without an entropy bonus, the policy entropy falls from 0.68, close to a uniform policy, to below 0.3 within the first 40,000 decisions, and the mean episode reward stops increasing at the same time. With an entropy coefficient of 0.2, the policy remains close to uniform during training, but its deterministic evaluation still accepts most feasible request.

The DQN agent reaches 40.84% on average. It rejects a small number of feasible requests and accepts slightly more requests in total, but its gain remains small. For comparison, in the main evaluation the RFP raises the service level over CI by 2.40 pp (RFP-CI) and 5.15 pp (RFP-SeS). RFP-CI accepts almost the same number of requests as the DQN agent but serves considerably more passengers. Neither agent learns to reject feasible requests to preserve capacity for better-fitting later requests, which is the behavior the displacement label encodes offline.

Two properties of the setting could explain this result. First, beneficial rejections pay off only later in the block, and with about 1,290 episodes per run, credit assignment remains delayed and diffuse. Second, coupled time windows make feasibility depend on the entire committed schedule, which leaves a narrow, state-dependent admissible action set at each epoch. Hildebrandt et al. (2026) identify this as a setting where RL struggles. The result is specific to sparse rural demand with rigid time windows: with denser demand, more episodes and less rigid temporal constraints, an experience-based policy may perform differently.

References

Agatz, N., Erera, A., Savelsbergh, M., Wang, X., 2012. Optimization for dynamic ride-sharing: a review. Eur. J. Oper. Res. 223 (2), 295–303.

Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M., 2019. Optuna: a next-generation hyperparameter optimization framework. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. ACM, pp. 2623–2631.

Alonso-Mora, J., Samaranayake, S., Wallar, A., Frazzoli, E., Rus, D., 2017a. On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. Proc. Natl. Acad. Sci. 114 (3), 462–467.

Alonso-Mora, J., Wallar, A., Rus, D., 2017b. Predictive routing for autonomous mobility-on-demand systems with ride-sharing. In: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, pp. 3583–3590.

Anzenhofer, F., Schmidbaur, S., Klein, R., Steinhardt, C., 2025. The potential of governmental regulation on shared mobility-on-demand systems. Transp. Res. A: Policy Pract. 192, 104360.

Attanasio, A., Cordeau, J.-F., Ghiani, G., Laporte, G., 2004. Parallel tabu search heuristics for the dynamic multi-vehicle dial-a-ride problem. Parall. Comput. 30 (3), 377–387.

Azadeh, S.S., Atasoy, B., Ben-Akiva, M.E., Bierlaire, M., Maknoon, M.Y., 2022. Choice-driven dial-a-ride problem for demand responsive mobility service. Transp. Res. B: Methodol. 161, 128–149.

Baty, L., Jungel, K., Klein, P.S., Parmentier, A., Schiffer, M., 2024. Combinatorial optimization-enriched machine learning to solve the dynamic vehicle routing problem with time windows. Transp. Sci. 58 (4), 708–725.

Bengio, Y., Lodi, A., Prouvost, A., 2021. Machine learning for combinatorial optimization: a methodological tour d'horizon. Eur. J. Oper. Res. 290 (2), 405–421.

Berbeglia, G., Cordeau, J.-F., Laporte, G., 2012. A hybrid tabu search and constraint programming algorithm for the dynamic dial-a-ride problem. INFORMS J. Comput. 24 (3), 343–355.

Breiman, L., 2001. Random forests. Mach. Learn. 45, 5–32.

Chen, T., Guestrin, C., 2016. XGBoost: a scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 785–794.

Chen, X., Ulmer, M.W., Thomas, B.W., 2022. Deep q-learning for same-day delivery with vehicles and drones. Eur. J. Oper. Res. 298 (3), 939–952.

Chollet, F., 2015. Keras. https://github.com/fchollet/keras.

Cordeau, J.-F., Laporte, G., 2003. The dial-a-ride problem (DARP): variants, modeling issues and algorithms. Q. J. Belg. Fr. Ital. Oper. Res. Soc. 1, 89–101.

Cordeau, J.-F., Laporte, G., 2007. The dial-a-ride problem: models and algorithms. Ann. Oper. Res. 153 (1), 29–46.

Coslovich, L., Pesenti, R., Ukovich, W., 2006. A two-phase insertion technique of unexpected customers for a dynamic dial-a-ride problem. Eur. J. Oper. Res. 175 (3), 1605–1615.

Coutinho, F.M., van Oort, N., Christoforou, Z., Alonso-González, M.J., Cats, O., Hoogendoorn, S., 2020. Impacts of replacing a fixed public transport line by a demand responsive transport system: case study of a rural area in Amsterdam. Res. Transp. Econ. 83, 100910.

Dong, X., Chow, J. Y.J., Waller, S.T., Rey, D., 2022. A chance-constrained dial-a-ride problem with utility-maximising demand and multiple pricing structures. Transp. Res. E: Logist. Transp. Rev. 158, 102601.

Estrada, M., Salanova, J.M., Medina-Tapia, M., Robusté, F., 2021. Operational cost and user performance analysis of on-demand bus and taxi systems. Transp. Lett. 13 (3), 229–242.

Fernández-Delgado, M., Cernadas, E., Barro, S., Amorim, D., 2014. Do we need hundreds of classifiers to solve real world classification problems? J. Mach. Learn. Res. 15 (1), 3133–3181.

Fleckenstein, D., Klein, R., Steinhardt, C., 2023. Recent advances in integrating demand management and vehicle routing: a methodological review. Eur. J. Oper. Res. 306 (2), 499–518.

Friedman, J.H., 2001. Greedy function approximation: a gradient boosting machine. Ann. Stat., 1189–1232.

Gaul, D., Klamroth, K., Pfeiffer, C., Stiglmayr, M., Schulz, A., 2025. A tight formulation for the dial-a-ride problem. Eur. J. Oper. Res. 321 (2), 363–382.

Gaul, D., Klamroth, K., Stiglmayr, M., 2022. Event-based MILP models for ridepooling applications. Eur. J. Oper. Res. 301 (3), 1048–1063.

Géron, A., 2022. Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow. O'Reilly Media, Inc.
Ghiani, G., Manni, A., Manni, E., 2022. A scalable anticipatory policy for the dynamic pickup and delivery problem. Comput. Oper. Res. 147, 105943.
Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep Learning. MIT press.
Goodson, J.C., Ohlmann, J.W., Thomas, B.W., 2013. Rollout policies for dynamic solutions to the multivehicle routing problem with stochastic demand and duration limits. Oper. Res. 61 (1), 138–154.
Gückel, J., Fontaine, P., 2025. Fast shapley value approximation through machine learning with application in routing problems. Networks 86 (4), 402–427.
Van der Hagen, L., Agatz, N., Spliet, R., Visser, T.R., Kok, L., 2024. Machine learning–based feasibility checks for dynamic time slot management. Transp. Sci. 58 (1), 94–109.
Hastie, T., Tibshirani, R., Friedman, J.H., 2009. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Vol. 2. Springer.
Hildebrandt, F.D., Bode, A., Ulmer, M.W., Mattfeld, D.C., 2026. Shaping decision models for stochastic dynamic optimization problems via reinforcement learning. Networks 88 (1), 138–157.
Ho, S.C., Szeto, W.Y., Kuo, Y.-H., Leung, J. M.Y., Petering, M., Tou, T. W.H., 2018. A survey of dial-a-ride problems: literature review and recent developments. Transp. Res. B: Methodol. 111, 395–421.
Ichoua, S., Gendreau, M., Potvin, J.-Y., 2006. Exploiting knowledge about future demands for real-time vehicle dispatching. Transp. Sci. 40 (2), 211–225.
Jaw, J.-J., Odoni, A.R., Psaraftis, H.N., Wilson, N. H.M., 1986. A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows. Transp. Res. B: Methodol. 20 (3), 243–257.
Jungel, K., Parmentier, A., Schiffer, M., Vidal, T., 2026. Learning-based online optimization for autonomous mobility-on-demand fleet control. INFORMS J. Comput. 38 (3), 745–765.
Klapp, M.A., Erera, A.L., Toriello, A., 2020. Request acceptance in same-day delivery. Transp. Res. E: Logist. Transp. Rev. 143, 102083.
Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. Adv. Neural Inf. Process. Syst. 25.
Larsen, E., Lachapelle, S., Bengio, Y., Frejinger, E., Lacoste-Julien, S., Lodi, A., 2022. Predicting tactical solutions to operational planning problems under imperfect information. INFORMS J. Comput. 34 (1), 227–242.
LeCun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. Nature 521 (7553), 436–444.
Liyanage, S., Dia, H., 2025. On-demand technologies for public transport: insights from a melbourne survey. IEEE Open J. Intell. Transp. Syst..
Lundberg, S.M., Lee, S.-I., 2017. A unified approach to interpreting model predictions. Adv. Neural Inf. Process. Syst. 30.
Luo, Q., Nagarajan, V., Sundt, A., Yin, Y., Vincent, J., Shahabi, M., 2023. Efficient algorithms for stochastic ride-pooling assignment with mixed fleets. Transp. Sci. 57 (4), 908–936.
Luo, Y., Schonfeld, P., 2011. Online rejected-reinsertion heuristics for dynamic multivehicle dial-a-ride problem. Transp. Res. Rec. 2218 (1), 59–67.
López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Birattari, M., Stützle, T., 2016. The irace package: iterated racing for automatic algorithm configuration. Oper. Res. Perspect. 3, 43–58.
McCullagh, P., 2019. Generalized Linear Models. Routledge.
Melo, S., Gomes, R., Abbasi, R., Arantes, A., 2024. Demand-responsive transport for urban mobility: integrating mobile data analytics to enhance public transportation systems. Sustainability 16 (11), 4367.
Merchan, D., Winkenbach, M., Snoeck, A., 2020. Quantifying the impact of urban road networks on the efficiency of local trips. Transp. Res. a: Policy Pract. 135, 38–62.
Molenbruch, Y., Braekers, K., Caris, A., 2017. Typology and literature review for dial-a-ride problems. Ann. Oper. Res. 259 (1), 295–325.
Nielsen, D., 2016. Tree Boosting with XGBoost-Why does XGBoost win "every" Machine Learning Competition? Master's thesis. NTNU.
Parragh, S.N., Doerner, K.F., Hartl, R.F., 2008. A survey on pickup and delivery problems: part 1: transportation between customers and depot. J. Betriebswirtsch. 58, 21–51.
Parragh, S.N., Pinho de Sousa, J., Almada-Lobo, B., 2015. The dial-a-ride problem with split requests and profits. Transp. Sci. 49 (2), 311–334.
Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al., 2011. Scikit-learn: machine learning in Python. J. Mach. Learn. Res. 12, 2825–2830.
Pina-Pardo, J.C., Silva, D.F., Smith, A.E., Gatica, R.A., 2024. Dynamic vehicle routing problem with drone resupply for same-day delivery. Transp. Res. C: Emerg. Technol. 162, 104611.
Porru, S., Misso, F.E., Pani, F.E., Repetto, C., 2020. Smart mobility and public transport: opportunities and challenges in rural and urban areas. J. Traffic Transp. Eng. 7 (1), 88–97.
Powell, W.B., 2007. Approximate Dynamic Programming: Solving the Curses of Dimensionality. Vol. 703. John Wiley & Sons.
Probst, P., Wright, M.N., Boulesteix, A.-L., 2019. Hyperparameters and tuning strategies for random forest. Wiley Interdiscip. Rev.: Data Min. Knowl. Discov. 9 (3), e1301.
Riedler, M., Raidl, G., 2018. Solving a selective dial-a-ride problem with logic-based benders decomposition. Comput. Oper. Res. 96, 30–54.
Rist, Y., Forbes, M., 2022. A column generation and combinatorial benders decomposition algorithm for the selective dial-a-ride-problem. Comput. Oper. Res. 140, 105649.
Ritzinger, U., Puchinger, J., Rudloff, C., Hartl, R.F., 2022. Comparison of anticipatory algorithms for a dial-a-ride problem. Eur. J. Oper. Res. 301 (2), 591–608.
Sadana, U., Chenreddy, A., Delage, E., Forel, A., Frejinger, E., Vidal, T., 2025. A survey of contextual optimization methods for decision-making under uncertainty. Eur. J. Oper. Res. 320 (2), 271–289.
Schilde, M., Doerner, K.F., Hartl, R.F., 2011. Metaheuristics for the dynamic stochastic dial-a-ride problem with expected return transports. Comput. Oper. Res. 38 (12), 1719–1730.
Soeffker, N., Ulmer, M.W., Mattfeld, D.C., 2022. Stochastic dynamic vehicle routing in the light of prescriptive analytics: a review. Eur. J. Oper. Res. 298 (3), 801–820.
Stallhofer, V., Parragh, S.N., 2025. Event-based models for the electric autonomous dial-a-ride problem. Transp. Res. C: Emerg. Technol. 171, 104896.
Sutskever, I., Martens, J., Dahl, G., Hinton, G., 2013. On the importance of initialization and momentum in deep learning. In: International Conference on Machine Learning. PMLR, pp. 1139–1147.
Ulmer, M.W., 2020. Dynamic pricing and routing for same-day delivery. Transp. Sci. 54 (4), 1016–1033.
Ulmer, M.W., Goodson, J.C., Mattfeld, D.C., Hennig, M., 2019. Offline–online approximate dynamic programming for dynamic vehicle routing with stochastic requests. Transp. Sci. 53 (1), 185–202.
Ulmer, M.W., Goodson, J.C., Mattfeld, D.C., Thomas, B.W., 2020. On modeling stochastic dynamic vehicle routing problems. EURO J. Transp. Logist. 9 (2), 100008.
Ulmer, M.W., Thomas, B.W., 2020. Meso-parametric value function approximation for dynamic customer acceptances in delivery routing. Eur. J. Oper. Res. 285 (1), 183–195.
Ulmer, M.W., Thomas, B.W., Campbell, A.M., Woyak, N., 2021. The restaurant meal delivery problem: dynamic pickup and delivery with deadlines and random ready times. Transp. Sci. 55 (1), 75–100.
Vallée, S., Oulamara, A., Cherif-Khettaf, W.R., 2020. New online reinsertion approaches for a dynamic dial-a-ride problem. J. Comput. Sci. 47, 101199.
Van Engelen, M., Cats, O., Post, H., Aardal, K., 2018. Enhancing flexible transport services with demand-anticipatory insertion heuristics. Transp. Res. E: Logist. Transp. Rev. 110, 110–121.
Vansteenwegen, P., Melis, L., Aktaş, D., Montenegro, B. D.G., Vieira, F.S., Sörensen, K., 2022. A survey on demand-responsive public bus systems. Transp. Res. C: Emerg. Technol. 137, 103573.
Voccia, S.A., Campbell, A.M., Thomas, B.W., 2019. The same-day delivery problem for online purchases. Transp. Sci. 53 (1), 167–184.
Xie, J., Liu, Y., Chen, N., 2023. Two-sided deep reinforcement learning for dynamic mobility-on-demand management with mixed autonomy. Transp. Sci. 57 (4), 1019–1046.
Yu, X., Shen, S., 2019. An integrated decomposition and approximate dynamic programming approach for on-demand ride pooling. IEEE Trans. Intell. Transp. Syst. 21 (9), 3811–3820.

Zhang, C., Bengio, S., Hardt, M., Recht, B., Vinyals, O., 2021. Understanding deep learning (still) requires rethinking generalization. Commun. ACM 64 (3), 107–115.
Zhang, J., Luo, K., Florio, A.M., Van Woensel, T., 2023. Solving large-scale dynamic vehicle routing problems with stochastic requests. Eur. J. Oper. Res. 306 (2), 596–614.
Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O., 2017. Proximal policy optimization algorithms. arXiv:1707.06347.