EURO Journal on Transportation and Logistics 12 (2023) 100104

Contents lists available at ScienceDirect

EURO Journal on Transportation and Logistics

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

Combining value function approximation and multiple scenario approach for the effective management of ride-hailing services

R.-Julius O. Heitmann [a,*], Ninja Soeffker [b], Marlin W. Ulmer [c], Dirk C. Mattfeld [a]

[a] Technische Universität Braunschweig, Decision Support Group, Germany
[b] Universität Wien, Department of Business Decisions and Analytics, Austria
[c] Otto von Guericke Universität Magdeburg, Chair of Management Science, Germany

ARTICLE INFO

Keywords:
Ride-sharing
Dial-a-ride
Dynamic vehicle routing
Value function approximation
Multiple scenario approach

ABSTRACT

The availability of various services for individual mobility is increasing, especially in urban areas. Dynamic ride-hailing services address these aspects and are gaining market share with providers such as MOIA, UberX Share, Sprinti or BerlKönig. To be able to offer competitive pricing for such a service and at the same time provide a high service quality (e.g. fast response times), effective capacity management is needed. In order to reach this goal, two challenges have to be met by the service provider. On the one hand, a proper demand control has to be installed, which optimizes the responses to transportation requests from customers. On the other hand, suitable fleet control needs to be set in place to optimize the route of the fleet so that the demand can be met. Papers in the literature do solve both but typically focus on one of these two challenges. As an example, value function approximation (VFA) can be used to learn a service offering decision while anticipating future incoming requests. A typical example of a routing-focused method is the multiple scenario approach (MSA) creating a routing which anticipates future requests using a sampling method. In this paper, we combine VFA and MSA to address the two challenges in an effective way. The resulting method is called anticipatory-routing-and-service-offering (ARS). We find that the combined method significantly outperforms the individual components, improving not only the total reward but also the accepted requests. It is found that this performance is particularly high with a heavy workload and thus resources are relatively scarce. We analyse how and under which conditions the components together or individually are particularly important.

1. Introduction

Mobility in urban areas has undergone a paradigm shift in recent decades, which can be attributed to the availability of digital mobility services, the increased demand for transportation and the broad availability of mobile devices which enables accessibility for mobility applications. Additionally an increase in the number of vehicles on the roads leads to congestion and thus to reduced service quality due to increased transport times. For the customer, on the other hand, mobility is perceived less as a tour with a specific vehicle but more as a transportation service between two locations independent of the modes of transportation (Forbes, 2020). This requirement abstracts the transport service and puts more focus on the overall quality of the service, which may be expressed in terms of availability, price or emissions, for example. To provide an according service for customers, a service provider may offer flexible rides via mobile application using a fleet of vehicles. To lower costs and increase efficiency, the rides may be shared with other customers. Such services can be summarized under the term dynamic dial-a-ride. A subset of the services are often referred to as ride-hailing services which focus more on substituting a public transport service. The ARS method presented in this paper aims to optimize a ride-hailing service as customer requests may be rejected. When implementing a dial-a-ride service for applications such as an medical or an elderly transportation service, a service has to be guaranteed and as a result a rejection may not be an option. In this case the medical transportation service would not be referred to as a ride-hailing service.

In a ride-hailing service like the one we are considering, customers can request transportation between two locations at their times of choice. When the service provider receives such a spontaneous request, two interrelated decisions have to be made. First, a decision has to be made whether service is offered to the customer. Second, a route for the fleet of vehicles has to be determined. In case of a positive response to the customer request, the provider generates revenue and needs to adapt the routing of the available fleet of vehicles such that all

* Corresponding author.
E-mail address: r.heitmann@tu-braunschweig.de (R.-J.O. Heitmann).

https://doi.org/10.1016/j.ejtl.2023.100104
Received 3 February 2022; Received in revised form 4 December 2022; Accepted 17 January 2023
Available online 24 January 2023

accepted requests can be serviced according to the service conditions. A vehicle may transport several customers at the same time. The goal and therefore the optimization task of the service provider is to maximize the accumulated received revenue over the service time horizon.

The two decisions have a short-term and a long-term effect. The acceptance of a request leads to an immediate reward, but also binds capacity, which may prevent reward in the long-term. Thus, with each decision, the service provider is confronted with a trade off between reward in the short run versus flexibility, which might manifest in reward in the long run. Efficiency and flexibility are influenced by both service offering and routing as the service offering determines if the capacity is used and the routing determines how the capacity is used. The interdependencies of the two decisions amplifies the complexity of the already challenging individual parts.

The majority of the literature focuses on one of the two components. The goal of this paper is to develop a method that integrates both the routing component as well as on the offering decision. To do so, the concept of a multiple scenario approach (MSA) from Bent and Van Hentenryck (2004) is combined with the concept of a value function approximation (VFA) approach as described in Powell (2021). The MSA approach focuses on the routing decision and offers service whenever possible. It uses sampled artificial requests which are assumed to follow the same distribution as future requests. A set of scenarios is created which contains the actual request as well as artificial requests. For this set, a routing solution is determined. The artificial requests are removed afterwards. Using a consensus mechanism, the most similar routing solution amongst the scenarios is selected for the current decision. The sampling strategy ensures that the routing anticipates the arrival of future requests and routes are set up accordingly. The VFA approach focuses on the service offering decision and approximates the expected future rewards for the two cases of offering service or not. For approximation, the VFA simulates a large number of service intervals, which may be hours, days or weeks and makes decisions based on the current approximation and updates the approximated values after each iteration. For the VFA in simulation and execution, a simple routing strategy is applied.

In this paper, we present an approach that integrates the MSA and VFA approaches. The VFA component replaces the consensus mechanism of the MSA component, enabling value-based scenario selection. We name the resulting algorithm anticipatory-routing-and-service-offering (ARS). In an extensive computational study, we show the superior performance of the ARS method compared to the individual components and benchmark policies. We analyse when the service offering and routing components are particularly important and how the components influence one another. It can be observed that with fewer incoming requests and therefore smaller degree of freedom for rejection, the routing component gets more important. Accordingly with more room for rejection the service offering component leads to an advantage. To our best knowledge we are the first to present an integrated approach with both a service offering and a routing decision focus. While the method is designed for ride-hailing service providers the general concept may be valuable for other on-demand routing services as well.

Section 2 first provides a classification of literature. The underlying problem definition and the model definition are formulated in Section 3. The solution methods considered are discussed in Section 4. Section 5 gives an overview of the instances used as well as the analysis of results of the experiments. An outlook is given in Section 6.

2. Literature review

In the following, an overview of the literature is given. In Section 2.1, we discuss ride-hailing, dial-a-ride and dynamic dial-a-ride problems in general. In a second step, we discuss literature in terms of their focus on action components, focusing on the routing decision in Section 2.2 or the service offering decision in Section 2.3. A classification of literature in a tabular form with a distinction on decision components is provided in Section 2.4.

2.1. Dial-a-ride-problem

The ARS method presented in this paper solves a dial-a-ride problem. The method allows for rejections of customer requests, which makes it especially suitable for use in ride-hailing services, which is a subset of dynamic dial-a-ride services. For other dynamic dial-a-ride services such as for example medical transport acceptance may have to be guaranteed. However, due to the similarity of the methods used in the literature, this chapter focuses mainly on dynamic dial-a-ride problems in general. For a better overview of static dial-a-ride problems, we refer to the literature review by Cordeau and Laporte (2007).

Dial-a-ride problems are involved in a variety of use cases which is reflected in the related literature. One of those use cases is health care, which includes for example mobility services for low mobility patients. Zhang et al. (2015) present an optimization of patient transportation using real world data from Hong Kong Hospital Authority. Due to the underlying ambulance routing application, the optimization of a dial-a-ride problem includes working periods and disinfection times.

Another common use case are shared mobility services. A fleet of vehicles is offered as shared resource from which rides between two locations at specific times can be requested. These rides may be provided as a shared service transporting more than one customer at a time. Marković et al. (2015) propose a method optimizing a dynamic dial-a-ride problem for a transportation service located in Maryland. Schilde et al. (2014) propose a set of heuristics optimizing a dynamic dial-a-ride problem. The used data is based on real world data from Vienna. Häll et al. (2015) analyse the impact of model parameters to derive guidelines to design dial-a-ride based services. The data used for this analysis is derived from a community transport system. Agatz et al. (2011) use a rolling horizon strategy to optimize a ride-sharing service by minimizing the travelled distance of vehicles and transportation costs for customers with an Atlanta area data set. Garaix et al. (2011) seek to maximize the passenger occupancy rate for a data set from a rural zone of France by the use of a column generation approach. Xing et al. (2009) create a ride-sharing concept for the area of Bremen, which is based on a multi-agent model. Masson et al. (2014) extend the dial-a-ride problem by the element of transfers, which enable the user to change rides during the trip.

While the approaches mentioned focus on the operational level of route planning, approaches such as fluid approximations or queueing approximations can also be found in the literature that refer to the tactical or strategic level of optimization. For an overview of the other levels of optimization, please refer to a literature review by Wu and Xu (2022). Since approaches of the other levels may be integrated, these approaches are considered complementary.

2.2. Literature focusing on dynamic routing optimization

Dynamic dial-a-ride belongs to the problem class of stochastic dynamic vehicle routing (SDVR) problems. One part of the SDVR literature focuses on the routing decision. Here, the service offering decision is generally positive as long as a feasible integration of the customer request into the route is possible. Papers which focus on this strategy mainly pursue an efficient strategy for route integration as more requests can be accepted this way.

A part of the approaches focused on routing use a sampling approach in which a representative sample of artificial requests is drawn to enable the anticipation of future requests. Bent and Van Hentenryck (2004) propose a multiple scenario approach, using a sampling method for customer requests, to create multiple scenarios containing routings which anticipate possible future requests. Schilde et al. (2014) combine and assess multiple solution approaches, which are partially used by the method presented in this paper as well for the routing strategy. In contrast they do not focus the service offering decision. With this, they optimize a medical transportation service for a Vienna data set. Ghiani et al. (2009) optimize a vehicle dispatching problem for a same-day courier service. They use a sampling strategy as the anticipatory component for the routing strategy based on a metric for flexibility in the near-future".

Furthermore, a variety of other solution approaches can be found in literature focused on the routing component, which use a variety of optimization strategies. Maalouf et al. (2014) include fuzzy logic in their heuristic as an anticipation component for route optimization. Mes et al. (2010) develop a method for a dynamic dial-a-ride problem including scheduling and pricing using dynamic programming following a look ahead strategy. Ferrucci and Bock (2014) present a tabu search based neighbourhood operator optimizing routing for dynamic dial-a-ride problems. Riley et al. (2020) optimize a mobility service for the New York City area. The proposed strategy is an integration of a dispatching strategy, a prediction of zone-to-zone demand and a relocation strategy. The focus of this work is on the relocation of large vehicle fleets. Pouls et al. (2020) propose a mixed-integer programming approach for a repositioning strategy, which involves a forecast component to enable a form of look ahead strategy. Pillac et al. (2018) propose a method for a dynamic routing and scheduling problem taking multiple resources into account as for example staff, skills, tools and spare parts. The methods discussed in this section focus on the routing decision, which distinguishes them from the method proposed in this paper that focuses on both decision components.

2.3. Literature focusing on service offering decision

The second decision component in SDVR problems concerns whether a service if offered or not for taking actions on requests are the decisions on whether a service is offered or not, which means whether the request is accepted or rejected. The goal of taking the decision with respect to future decisions can be achieved by anticipation of the stochastic process using either a specific set of rules, sampling strategies or learning a value function from historic data (VFA). Methods that allow to reject feasible requests typically involve simpler routing components like cheapest insertion. Ulmer et al. (2018) use a VFA approach which has the beneficial property that the time effort for a single decision is relatively small as the offline learning effort can be processed beforehand. A further VFA approach is proposed by Yu and Shen (2019), who optimize traffic for New York Taxi data for car pooling up to two customers. Ulmer et al. (2019b) use a VFA approach to optimize a same-day delivery problem. To increase the performance of the approach, they involve an anticipatory depot return strategy using a simple insertion heuristic as routing component. Ulmer et al. (2019a) embeds a VFA in a look ahead method to capture temporal and spatial information when deciding about offering service to a requesting customer. However, both components focus on the decision whom to offer service to. For routing, a simple heuristic is applied. Shah et al. (2020) use a VFA approach for relocating vehicles in the city. As this approach combines the service offering decision with a routing decision in form of relocation, this paper represents one of the few identified works which focus on both parts of the action. Kullman et al. (2022) also choose a deep learning strategy for a q-learning approach to optimize a taxi dispatcher for a Manhattan data set. Agussurja et al. (2019) propose estimation strategies with representative states on a discretized state space for a Singapore data set for a last-mile transport service. Ulmer and Thomas (2020) propose an VFA method using a combination of a linear function and a look-up-table as value function to take advantage of both methods. In contrast to the method proposed in this paper, the methods in this section focus on the service offering decision instead of focusing on both decision components.

Table 1
Classification of literature.

Literature	Routing	Service off.	Sol. method
Bent and Van Hentenryck (2004)	x	–	LA
Hvattum et al. (2007)	x	–	LA
Pureza and Laporte (2008)	x	–	PFA
Cortés et al. (2009)	x	–	LA
Ghiani et al. (2009)	x	–	LA
Xing et al. (2009)	–	x	RO
Beaudry et al. (2010)	x	–	RO
Mes et al. (2010)	–	x	VFA
Agatz et al. (2011)	x	–	RO
Garaix et al. (2011)	x	–	RO
Meisel (2011)	–	x	VFA
Azi et al. (2012)	x	–	LA
Sheridan et al. (2013)	x	–	PFA
Ferrucci and Bock (2014)	x	–	LA
Ehmke and Campbell (2014)	–	x	LA
Schilde et al. (2014)	x	–	LA
Toriello et al. (2014)	x	–	VFA
de Armas and Melián-Batista (2015)	x	–	RO
Sarasola et al. (2016)	x	–	LA
Ferrucci and Bock (2016)	x	–	LA
Kuo et al. (2016)	x	x	RO
Srour et al. (2018)	x	–	LA
Pillac et al. (2018)	x	–	RO
Ulmer et al. (2018)	–	x	VFA
Zhang et al. (2018)	x	–	LA
Agussurja et al. (2019)	–	x	VFA
Ulmer et al. (2019a)	–	x	LA+VFA
Ulmer et al. (2019b)	–	x	VFA
Yu and Shen (2019)	–	x	VFA
Shah et al. (2020)	x	x	VFA
Pouls et al. (2020)	x	–	LA
Riley et al. (2020)	x	–	LA
Ulmer and Thomas (2020)	–	x	LA
Kullman et al. (2022)	–	x	VFA
Our work	x	x	LA+VFA

2.4. Classification of literature

Table 1 present a selection of papers which are discussed in terms of the main strategies for routing and service offering. There are different types of solution approaches in the literature, where it is noticeable that certain types of solution approaches mostly focus more on specific parts of the action. The table shows papers from the entire SDVR-literature. The columns *routing* and *service offering* indicate papers which focus on the respective parts. The column solution method indicates the methodology used according to the classification of Ulmer et al. (2020) which distinguishes between the four categories reoptimization strategies (RO), policy function approximations (PFA), look ahead strategies (LA) and value function approximations (VFA). RO strategies optimize a system with no anticipation of the stochastic process as they only take the current point in time into account. PFAs are constructed using a specific set of rules to anticipate the stochastic process. Those rules typically reflect specific strategies which are based on real life strategies like a rule of thumb or waiting rules. The other strategies involve an explicit anticipation of the stochastic process, which is sampling of artificial requests for LA and simulation for VFA strategies. MSA strategies belong to the LA category. VFAs approximate the expected future reward for a decision in a state, which contains all information at the current point in time necessary to make a decision. To the best of our knowledge we are the first to present a method that contains a LA and VFA component, which results in a double focus on routing and service offering. With this, we are able to observe not only the improvements of the individual components but also analyse the interdependencies. As Kuo et al. (2016) also focus on both routing and service offering a closer comparison is necessary. In contrast to the algorithm proposed in this paper a re-optimization approach is used, which does not include an anticipatory component.

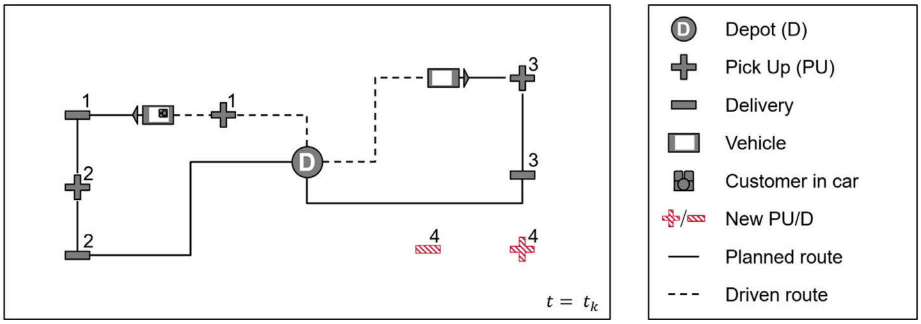

Fig. 1. Example of a state.

3. Problem definition

In the following, a problem description is given in Section 3.1 and the related model is presented in Section 3.2.

3.1. Problem description

In a dynamic dial-a-ride problem, customers request a transportation from the provider, which occur from perspective of the service provider randomly at times t_k during the service time T. The customer requests to be transported between two locations. the customer needs to be picked up from location l_k^p at time t_k^p or later but not before that time and dropped at location l_k^d at time t_k^d or earlier but not later then that time. The customer is willing to share the ride with other customers. The request r_k may be described as a tuple

$$r_k = (t_k, t_k^p, l_k^p, t_k^d, l_k^d). \tag{1}$$

A service provider operates a fleet V of vehicles. Each vehicle can transport $c_{\max}$ customers at the same time. Travelling between two locations l_i and l_j covers the distance $\delta(l_i, l_j)$ and takes the time $\tau(l_i, l_j)$. As the provider receives the request r_k in state s_k an action x_k has to be chosen, consisting on the one hand of the service offering decision d_k, meaning the decision to accept or reject the request, and the routing decision θ_k, meaning how the accepted request is integrated into a route planning for the fleet. A request may be feasibly integrated into the route if, for all requests accepted by the service provider, a vehicle $v_m \in V$ picks up the respective customer at the agreed location l_k^p on or after the agreed time t_k^p and arrives at the delivery point l_k^d before or on the agreed time t_k^d. If a feasible route is available, the service provider may offer service to the customer. Else, the service offering decision is reject in any case. For an accepted request, the service provider receives a transportation reward R from the customer. The service provider's goal is to maximize the accumulated rewards received by the customers over the service time horizon.

3.2. Model definition

In this section, the model of the dynamic dial-a-ride problem is provided. The problem is modelled using a Markov Decision Process (MDP). The components of the MDP are described in the following.

State s_k

A system is in a state s_k which occurs in decision epoch k when a new request r_k is received by the service provider. At the current point in time $t_k \in [0, T]$ the vehicles $v_m \in V$ drive on the route

$$\theta_{mk} = \{(l_1^v, t_1^v), \dots, (l_{n_{mk}}^v, t_{n_{mk}}^v)\}, \tag{2}$$

which contains the n_{mk} ordered waypoints, which are defined by locations and times. The route includes both the past waypoints to include information on clients already picked up but not yet dropped off and the planned future waypoints. At epoch k the vehicles are positioned at locations l_{mk}^v. The union of vehicles' locations is defined as l_k^v and the union of the vehicle routes as θ_k.

The set B_k describes the set of all already accepted requests at time t_k. A state s_k can thus be described as a tuple

$$s_k = (t_k, l_k^v, \theta_k, r_k, B_k). \tag{3}$$

Fig. 1 shows an example of a state. Two vehicles move along routes to fulfil already accepted pick up assignments, shown as plus-symbol, and delivery assignments, shown as minus-symbol. Both vehicles are already on a route, where the left car has to serve request 1 and 2 and the right car serves request 3. The left vehicle has already picked up its customer while the right vehicle is still on the way to a pick up location. A new request 4, shown in red, is received.

We assume that customers request service spontaneously. In Appendix D, we show that our methodology also works in case customer preannounce their request some time ahead of service.

Action x_k

Next, an action x_k containing two parts has to be chosen. First, a decision $d_k \in D(s_k)$ must be made about whether to offer service for the received request r_k, which is reflected by a value of 1 for acceptance and 0 for rejection of the request:

$$d_k = \begin{cases} 1 & \textit{if service is offered} \\ 0 & \textit{if service is not offered} \end{cases} \tag{4}$$

With the acceptance of the request r_k, a new set of accepted requests $B_k^x = B_k \cup r_k$ results. In case of rejection $B_k^x = B_k$. A feasible route plan $\theta_k^x \in \Theta(s_k, d_k)$ has to be selected to serve the accepted requests B_k^x, where $\Theta(s_k, d_k)$ represents the set of all feasible routes in the state s_k for decision d_k. A route plan is feasible as long as for every $r_i \in B_k^x$, the customer is picked up after the earliest requested pick up time t_i^p in location l_i^p and dropped off before the latest drop off time t_i^d in location l_i^d. Here, the action x_k can therefore be represented as a tuple:

$$x_k = (d_k, \theta_k^x) \tag{5}$$

For the action x_k, a reward $R(s_k, x_k)$ is received. The reward is in some way dependent on the service provided and an agreed fee for a transport ϕ.

$$R(s_k, x_k) = \phi(r_k) \tag{6}$$

Fig. 2 shows options for actions in the state s_k. In case of rejection of the customer request, no service is offered to the customer. As a result, the route of the fleet may stay unchanged and no immediate reward is gained as the customer does not pay for the transport, which is shown as Option 1. Both of the acceptance actions, which are presented in Option 2 and 3, would result in the same reward as the requested direct transportation distance for the new customer would be the same. One aspect which may be considered when accepting the request with the left car is the resulting route for the fleet as both cars would end up more focused on the right side of the map. This may result in inability to accept request on the left side of the map in the future.

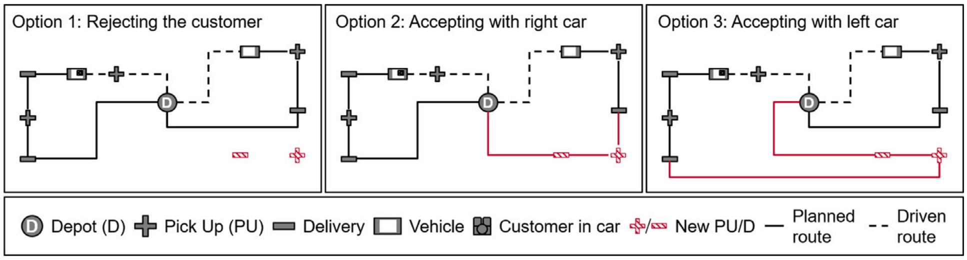

Fig. 2. Routing options to serve the request.

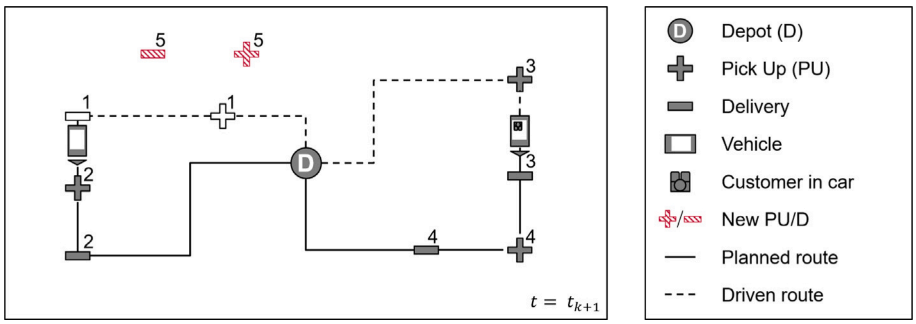

Fig. 3. Realization of random process resulting in the state of next epoch.

Post-decision state s_k^x

The combination of state s_k and action x_k leads to a post-decision state s_k^x. As discussed in Powell (2021), this has the purpose of obtaining a deterministic state between decision and the subsequent random process ω_{k+1}. The post-decision state s_k^x therefore results from the state s_k and the action x_k and may be represented in a simple form as the tuple

$$s_k^x = (s_k, x_k). \tag{7}$$

In order to represent the post-decision state in a similar form to the state and to make the information contained better visible, it can be presented in a more detailed but equivalent form as tuple

$$s_k^x = (t_k, l_k^v, \theta_k^x, d_k, r_k, B_k^x). \tag{8}$$

In the given example, we assume that the request was accepted by the right vehicle (Option 2 in Fig. 2).

Random information ω_{k+1}

After the post-decision state s_k^x, a realization of the random process $\omega_{k+1} \in \Omega$ takes place. There are two possible types of realizations. First, no request may occur in the remainder of the horizon, which means an empty realization $\emptyset \subset \Omega$, terminating the decision process. Second, a new request r_{k+1} is received, so that the random process takes place in the time period $[t_k, t_{k+1}[$. During this time, the vehicles in the fleet follow their routes until t_{k+1}, which leads to a change in the location of the vehicles and the customers still to serve. Let B_{k+1}^S be the set of requests which are served by the fleet during the random process in epoch $k+1$, which can be removed from the set of accepted requests $B_{k+1} = B_k^x \setminus B_{k+1}^S$. The result is a next state s_{k+1}.

Fig. 3 shows an example of a new request 5 being received. Request 1 is already served at the end of realization ω_{k+1}, which is why it is represented in white as it is not part of the state any more.

Optimal solution: policy π

A policy π is a function, which maps an unambiguous action x_k to every state s_k. The goal is to find the optimal policy π^* for which applies

$$R^*(\pi^*) = \max_{\pi \in \Pi} \mathbb{E} \sum_K R(s_k, x_k^\pi) | s_0. \tag{9}$$

Given the starting state s_0, the optimal policy π^* is the policy out of all policies Π which leads to the optimal reward $R^*(\pi^*)$. In case of the ride sharing on-demand mobility service, the policy is the optimizing component which responds to customer requests maximizing the reward function of the service provider.

4. Solution method

In the following, we present our method. A motivation for the presented method is given in Section 4.1. The individual components are introduced in Sections 4.2 and 4.3 as well as their integration in Section 4.4. The cheapest insertion heuristic, which is the basis for routing optimization in all implemented algorithms, is presented in Section 4.5.

4.1. Motivation

Decision-making for the problem contains two components, service offering and routing. Both pose challenges in themselves. Offering service now generates revenue but may lead to limited resources and inflexibility in the future. Routing decisions should carefully balance the efficient use of vehicle resources and therefore short travel times with the fleet's availability in time and space and its flexibility to integrate new requests in their routes. Furthermore, these two challenges in the individual decision components are intertwined.

As shown in the literature review, the vast body of literature focuses on only one of the components, leaving the other to a simple heuristic mechanism. In this work, we propose an integrated consideration of both components. To this end, we combine two existing methods from the literature, a MSA and a VFA. The resulting algorithm is called the anticipatory-routing-and-service-offering (ARS) algorithm. First, the individual methods are introduced in more detail with MSA in Section 4.2 and VFA in Section 4.3. Second, their integration as ARS is presented in Section 4.4.

4.2. Routing component

The routing component is based on the approach of Bent and Van Hentenryck (2004), who proposed a multiple scenario approach

(MSA). We describe the approach verbally before providing a formal description in the form of a pseudocode. Within this description, the elements of the pseudocode are introduced.

For epoch k, a state s_k is given. The decision to insert the request r_k should also anticipate the stochastic process. For this purpose, a set of possible scenarios is created. For each of these scenarios, a set of artificial requests from a set of historic requests is sampled (*sampleFutureRequests*). First the request r_k and then the artificial requests are integrated into the route as long as a feasible route with artificial requests can be created (*integrateRequest*). There is a route for each scenario, which includes the integrated artificial requests and thus anticipates the stochastic process to some degree. The artificial requests are removed and feasible routes without artificial requests are obtained (*removeArtificials*). At this point, one of the feasible routes is selected using a consensus function. In Bent and Van Hentenryck (2004), this consensus is a similarity function. The route that is most similar to all routes of the other scenarios is selected. The selected route is chosen as routing θ_k^x subsequent to the decision.

Bent and Van Hentenryck (2004) use a variable neighbourhood search (VNS) to create the routing. The request is always accepted if a feasible solution is available. In the ARS, a cheapest insertion is used instead of a VNS, which integrates the request into the route at the feasible location minimizing the related detour. The cheapest insertion heuristic is described in more detail in Section 4.5. To improve the performance of the routing, a k-Opt procedure is added similar to Schilde et al. (2014). The resulting route construction method of cheapest insertion and k-Opt is referenced as *integrateRequest* in the following.

Instead of one feasible solution, a set of feasible solutions is maintained at one point in time. Solutions that became infeasible are sorted out. Since only one routing can be executed at the time of decision, one solution is selected. A consensus function selects the solution that is most similar to all other maintained solutions, which is defined as

$$f_k(\theta) = \sum_I M_k[n_i, succ(\theta_k, LDC(n_i)].\tag{10}$$

M_k denotes a two-dimensional matrix containing the number of plans in all maintained plans at epoch k in which the vehicle n_i with index $i \in I$ departs from the specific next customer. The respective succession of waypoints from the current point in time on is described as $succ(\theta, LDC(n_i))$ with $LDC(n_i)$ as the last customer from which the vehicle departed in the plan execution.

In order to anticipate the stochastic request process for the routing decision, historical request data is used. Assuming the historical requests follow the same distribution as the stochastic request process, a sample is drawn from the historical requests and fed to a planning scenario as artificial requests. A set of m scenarios is created from which the artificial requests are removed after the routes are generated. Using the consensus function (*consensusSelection*), one scenario is then selected which gives the action x_k. To facilitate understanding, Algorithm 1 states the approach as pseudo code.

Algorithm 1: Multiple scenario approach: Deciding on action x_k

```
Result: Decision on action x_k = (d_k, θ_k^x) given state s_k and set of
        historic requests R_h
θ'_k =integrateRequest(θ_k, r_k);
if r_k could not be inserted into feasible route then
    d_k = 0
    return(d_k, θ_k);
end
scenarioRoutes Θ_k^scen = emptyArray();
for i ← 1 to numberScenarios by +1 do
    artificialRequests R_h^i = sampleFutureRequests(R_h,
     numberArtificials);
    artificialRoute θ_k^{art_i} = integrateRequest(θ'_k, R_h^i);
    scenarioRoute θ_k^{scen_i} = removeArtificials(θ_k^{art_i}, R_h^i);
    Θ_k^scen add θ_k^{scen_i};
end
selectedRoute θ_k^select = consensusSelection(Θ_k^scen);
d_k = 1
return(d_k, θ_k^select);
```

4.3. Service offering decision

As a methodology for making a decision on the acceptance of a request r_k, value function approximation (VFA) is discussed in the following, with reference to Powell (2021). The value of a state $V(s_k)$ is introduced which is the expected future value from accepted requests. The decision in routing focused methods is mostly acceptance, as long as the request can be feasibly integrated into the route. However, such a positive acceptance strategy can lead to resources being occupied in the future, which prevents future acceptance. A higher utilization of resources would thus make acceptances in the future less likely.

The strategy of approximate dynamic programming is to approximate the value of a state $V(s_k)$, which reflects the expected future rewards of the state. The basis for this is the Bellmann equation, which describes the value of the state using the immediate reward and the expected value of the resulting state given a policy π:

$$V^\pi(s_k) = \max_{x_k}(R(s_k, x_k) + \mathbb{E}\{V(s_{k+1})|s_k^x\}).\tag{11}$$

The equation contains an expectation of the value $V(s_{k+1})$ as the random process ω_{k+1} takes place in between, which makes the approximation of a future value $V(s_{k+1})$ difficult. To learn the expected value for a state, the post-decision state s_k^x is introduced, which is the state after the decision and before the stochastic process, so that the equation can be written as

$$V^\pi(s_k) = \max_{x_k}(R(s_k, x_k) + V^\pi(s_k^x)).\tag{12}$$

This value function of the post-decision state $V(s_k^x)$ is initialized with a defined value. It is updated after every ending of time horizon T. Thereby, the value function approximates the expected future reward following the post-decision state. To make a decision using VFA, the request r_k is first integrated into the route θ_k^x. Therefore, the algorithm can decide between two options with the current route for rejection and the route with the integrated request for acceptance. If no feasible route exists that includes the request, the request is rejected and the current route is still feasible. If a feasible route exists, the reward $R(s_k, x_k)$ is determined. Both decision options are assessed in terms of immediate reward and expected value of the resulting post-decision state. The option with the higher sum of immediate reward and value of the post-decision state is selected. To facilitate understanding of the algorithm, Algorithm 2 gives the pseudo code of the algorithm.

There are several options of value functions. The most common ones in literature are the linear model or the look up table. For this paper, the look up table is chosen. To cope with the dimensionality of a state, a set of state variables is defined, which define a distinct state and can be used to approximate the value of a post-decision-state s_k^x. For the tabular form, the continuous variables need to be discretized. This step is called aggregation. As a result, two states are identical for the approximated value function as long as the aggregated values for the defined state variables are identical. For aggregated states, a value is created in the table, which is initially assigned a defined initial value.

Ulmer et al. (2020) presented the two state variables time and free time budget, which describe the current point in time and the free time of vehicles which is currently not booked on routes. The algorithm therefore gets information on how much time is left and information on flexibility in the form of free time resources. For this paper, we introduce an additional state variable regarding flexibility in

Algorithm 2: Value function approximation: Deciding on action x_k

Result: Decision on action $x_k = (d_k, \theta_k^x)$ given state s_k, value function $V^{\pi_i}(s_k^x)$ with current policy $\pi_i \in \Pi$ and reward function $R(s_k, x_k)$

postDecisionStateRejection $s_k^{rej} \leftarrow s_k$;
routingAcceptingRequest $\theta_{k+1} =$ *integrateRequest*(θ_k, r_k);
if *r_k could be inserted into feasible route* **then**
  postDecisionStateAcceptance $s_k^{acc} = (t_k, l_k, \theta_{k+1}, r_k, B \cup r_k)$
else
  $d_k = 0$
  return(d_k, θ_k);
end
if $V^{\pi_i}(s_k^{acc}) - V^{\pi_i}(s_k^{rej}) + R(s_k^{acc}, (d_k = 1, \theta_k^x)) - R(s_k^{rej}, (d_k = 0, \theta_k)) \geq 0$ **then**
  $d_k = 1$
  return(d_k, θ_k^x);
else
  $d_k = 0$
  return(d_k, θ_k);
end

the form of capacities. As a result, the three following state variables are introduced:

time: The current time t_k of the state.

time on route: The planned time spent by all vehicles of the fleet V outside the depot is described as **time on route**. It is the sum of both driving time and waiting time outside the depot after current point in time t_k. With a route $\theta_k = ((l_1, t_1), \ldots, (l_\beta, t_\beta))$ and the depot location l^D, the **time on route** can be defined as

$$tor = \sum_{i=1}^{\beta-1} t_{i+1} - t_i | l_i \neq l^D \vee l_{i+1} \neq l^D, t > t_k. \tag{13}$$

All periods of time after the current point in time are summed up, if those periods do not both start and end in the depot, which would be waiting time at the depot. The **time on route** state variable is therefore comparable to the free time budget state variable (Ulmer et al., 2020).

load factor: The sum of the planned capacity utilization of the vehicle fleet V after the current time t_k is defined as the **load factor**. The state variable thus describes the planned temporal occupation of vehicle seats. With the set of all requests B_k not yet completely served at time t_k and the planned route θ_k, a set can be defined in which all requests $r_i \in B_k$ are assigned the planned times of pick up t_i^{pick} and drop offs t_i^{drop} in the route θ_k. The loa can thus be defined as

$$loa = \sum_{B_k} t_i^{drop} - t_i^{pick} | t > t_k \tag{14}$$

In the context of this work, one request transports exactly one customer. If multiple persons can be included in one request, the loa can be extended by an additional factor.

The state variables presented contain explicit temporal information and implicit spatial information, as temporal distances are dependent on local distances. Explicit geographic modelling of state variables is avoided, as it turned out to be less useful in the context of other research, such as in the case of Ulmer et al. (2018).

In order to make the features graphically understandable, Fig. 4 presents an exemplary vehicle to which three requests are allocated. The current time t_k and thus the feature **time** is displayed as a vertical blue dashed line with a blue triangle on the bottom as an indicator. Starting from the current point in time, the vehicle serves the requests displayed as red bars. The feature **load factor** is therefore represented by the accumulated red striped bars. As trips to and from the customers are necessary for a feasible route, any trips required to and from the depot are shown in grey stripes. The feature **time on route** is the total time starting from time t_k at which the vehicle is on a route outside the depot which can be displayed using the green bars.

The values of post-decision states are learned by means of offline simulation. After each simulation run i, the values $V^{\pi_{i-1}}(s_k^x)$ are updated with the realized values $V_{\omega^i}(s_k^x)$ of the simulation run i using a learning rate α as shown in Eq. (15).

$$V^{\pi_i}(s_k^x) = (1-\alpha) V^{\pi_{i-1}}(s_k^x) + \alpha V_{\omega^i}(s_k^x) \tag{15}$$

Starting with the last realization of the simulation run, the rewards of the decisions are summed up backwards to determine the actual future accumulated reward at that time for the respective post-decision states. The resulting pairs of post-decision states and values are used to update the value function as the weighted sum of the previous value and the new value given the learning rate α. The procedure is formulated in Algorithm 3.

Algorithm 3: Value function approximation: Value function $V^\pi(S_k^x)$ update

Result: Update of the value function $V^{\pi_{i-1}}(s_k^x)$ given all realized post-decision states S_i^x, rewards $R(s_k, x_k)$ of all decision epochs K of simulation run i with learning rate α

backwardRewardSum $\rho \leftarrow 0$;
for k **in** *K decreasingly sorted* **do**
  $V^{\pi_i}(s_k^x) = (1-\alpha) V^{\pi_{i-1}}(s_k^x) + \alpha\rho$;
  $\rho \leftarrow \rho + R(s_k, x_k)$;
end

4.4. Integration of routing and decision component

The objective of this work is to integrate the MSA and VFA methods resulting in the anticipatory-routing-and-service-offering algorithm (ARS). The consensus function of the MSA method, which selects the scenarios on the basis of a decision rule, chosen as the integration point of the two methods. This selection will be learned in the integrated method, using the VFA component as the consensus function.

The ARS method initially follows the same steps as the MSA method. Artificial requests are sampled from historical requests (*sampleFutureRequests*). Next, the new request r_k and subsequently the artificial requests are inserted into the routes of the scenarios (*integrateRequests*) before removing the artificial requests from the route afterwards (*removeArtificials*). From the resulting set of scenarios, one needs to be selected for the decision x_k. To do this, the post-decision state is first determined for each of the scenarios.

For the selection of scenarios, the VFA approach is used and thus the MSA consensus function is replaced by the Bellman equation that compares the available options. For every scenario, the immediate reward is evaluated as the result of the related service offering decision. The expected future reward is approximated by the value function for the post-decision state resulting from the scenario. The scenario and the related route plan with the highest sum of immediate and expected future rewards is chosen which prescribes the action x_k for the epoch k. If two scenarios should be assessed as equally valuable, the last of the evaluated scenarios is chosen in our implementation, as a tiebreaker. The MSA approach keeps the scenarios created and uses them in future decisions. The implementation of the ARS algorithm used here discards the created scenarios after the decision. The creation of the scenarios at the time of the decision is convenient from an implementation perspective. The different artificial requests and the k-opt process used nevertheless lead to a high diversity of scenarios.

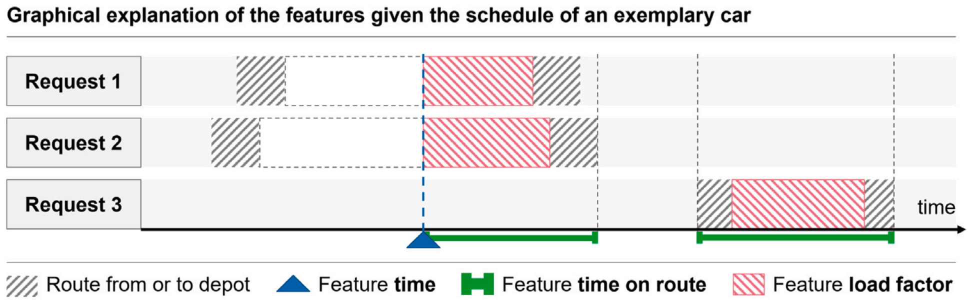

Fig. 4. Graphical representation of the features.

For the VFA component, the decision process does not change, but the number of decision options changes as a selection between multiple scenarios and a rejection is possible instead of the acceptance-or-rejection decision. The pseudo code for the resulting ARS algorithm is shown in Algorithm 4.

Algorithm 4: ARS: Deciding on action x_k

Result: Decision on action $x_k = (d_k, \theta_k^x)$ given state s_k and set of historic requests R_h

$\theta'_k = integrateRequest(\theta_k, r_k)$;
if *r_k could not be inserted in feasible route* **then**
  $d_k = 0$
  return(d_k, θ_k);
end
scenarioRoutes Θ_k^{scen};
for $i \leftarrow 1$ **to** *amountScenarios* **by** $+1$ **do**
  artificialRequests R_h^i = sampleFutureRequests(R_h, amountArtificials);
  artificialRoute $\theta_k^{art_i}$ = integrateRequest(θ'_k, R_h^i);
  scenarioRoute $\theta_k^{scen_i}$ = removeArtificials($\theta_k^{art_i}$, R_h^i);
  Θ_k^{scen} **add** $\theta_k^{scen_i}$;
end
postDecisionStates $S_k^{pds} \leftarrow$ getPostDecisionStates(t_k, l_k, Θ_k^{scen}, r_k, $B \cup r_k$);
postDecisionStateRejection $s_k^{rej} \leftarrow s_k$;
S_k^{pds} **add** s_k^{rej};
bestValue $v_{best} \leftarrow 0$;
selectedPostDecisionState s_k^{select};
for *scenarioState* s_k^{pds} **in** S_k^{pds} **do**
  if $R(s_k^{pds}, x_k^{pds}) + V^{\pi_i}(s_k^{pds}) - v_{best} \geq 0$ **then**
    $v_{best} \leftarrow R(s_k^{pds}, x_k^{pds}) + V^{\pi_i}(s_k^{pds})$;
    $s_k^{select} \leftarrow s_k^{pds}$
  end
end
return$(d_k^{select}, \theta_k^{select})$;

4.5. Cheapest insertion heuristic

Vehicle routing problems are NP-hard (Laporte and Nobert, 1987). Calculating the optimal solution of VRP problems thus becomes intractable for larger instances. Instead, heuristics are chosen that sensibly search the solution space for good solutions. Here, a cheapest insertion (CI) heuristic is chosen for the creation of routes for all described approaches. With this heuristic, a new waypoint (l^{new}, t^{new}) is inserted into the existing route

$$\theta^{old} = ((l_1, t_1), \dots, (l_\beta, t_\beta)) \tag{16}$$

at the point i so that the resulting route

$$\theta^{new} = ((l_1, t'_1), \dots, (l^{new}, t'^{new}), \dots, (l_{\beta+1}, t'_{\beta+1})) \tag{17}$$

results in the smallest detour, which can be represented as

$$i = \arg\min \sum_{j=1}^{\beta} \delta(l_j, l_{j+1}) \tag{18}$$

using the distance metric $\delta(l_i, l_j)$ for the distance between two locations. In a dial-a-ride problem, pick-up and a delivery waypoint must be integrated into the same route. In our CI implementation, we select the shortest distance detour. Since the vehicles start at the depot and return the depot at the end of the service time, the inserted waypoint can never be the first or last waypoint.

5. Computational experiments

In this section, the performance of the proposed approach is evaluated. For this purpose, instances based on taxi transport data from the New York region are constructed. The data basis as well as the derivation of the instances are discussed in Section 5.1. Section 5.2 examines the performance of the proposed approach. First, the overall performance compared to the single components is discussed. Subsequently, the influence of simulation parameters as well as the interaction of the two method components are examined in detail. This involves analyses on the advantages of the ARS components dependent on the problem. The efficiency of the decision and the learned selection of the method are then examined in more detail.

5.1. Simulation data

The instances for the simulation are created from real data from TLC Trip Record Data of the NYC Taxi and Limousine Commission 2018. This data set contains transport data of yellow and green taxis from the New York region, which are available open source. From 2009 to 2019, which excludes effects of the COVID-19 pandemic, information on customer transportation is provided, including pick up and delivery locations and times, as well as location and time points during the tours. In addition, speeds and total transport times are given.

To obtain a constant distribution of daily trips, we focus on weekdays only for our computational study. This approach does not lead to loss of generality as further models can be trained for weekends and holidays by using the respective data.

To allow for our comprehensive analyses, we focus on the hour around noon of the workdays. To show that ARS also outperforms the other methods on larger time frames, one exemplary experiment was carried out on a four hour time horizon which can be seen in Appendix A.

In order to create instances for the simulation from these data points, the information that must be available for a request r_k is first recalled:

$$r_k = (t_k, t_k^p, l_k^p, t_k^d, l_k^d) \tag{19}$$

To get pick up and delivery times and locations, the first and last points of the routes of the data points are selected. No attribute can

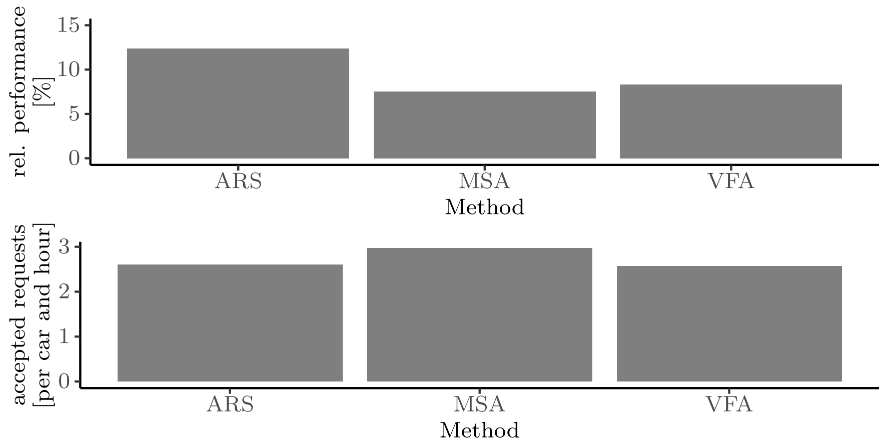

Fig. 5. Overall performance of ARS and benchmark approaches.

be identified in the TLC data for request times. Therefore, a random time before the trip start and within the interval is generated, which is selected as the time of receipt t_k. During the analysis of the routes, that is, the sequence of route points and times, it is noticeable that the trips are longer than the Manhattan distance, which is possibly a Manhattan distance with detours. Thus, a 0.5-norm is chosen as the distance metric between two points. For a simulation run, a subset of requests is randomly selected with 10 expected requests per hour unless stated otherwise.

In this paper, the reward function $R(s_k, x_k)$ is defined as linear dependent on the requested transportation distance, which reflects the idea that customers pay a fee for directly transported distances.

$$R(s_k, x_k) = \begin{cases} \delta(l_k^p, l_k^d) & if\ d_k = 1 \\ 0 & if\ d_k = 0. \end{cases} \quad (20)$$

The function thus represents the optimization of the revenues of a transport service. We acknowledge that such a reward favours the acceptance of high-value transports. An alternative choice of the reward function could be the amount of accepted requests, which in turn would most likely prefer short transports. The choice of the objective function must be made according to the specific use case or business goal. In order to test the behaviour of the method under a different reward model, experiments with an alternative reward function were carried out. The results for those experiments are reported in Appendix C.

The distance $\delta(l_k^p, l_k^d)$ may be specified in kilometres or equivalent and the factor $\phi(r_k)$ for cost per distance transported may depend on the times or locations of the request r_k. Unless otherwise stated, a fleet size of two vehicles is assumed, which can transport up to three customers at the same time. A constant travel speed is assumed for all vehicles. For the MSA and ARS method, three scenarios are computed. Simulations are carried out up to a maximum of 500.000 runs.

5.2. Results

In this section, the performance of the integrated ARS compared to the MSA and VFA methods is examined. As a benchmark, the cheapest insertion (CI) heuristic will be used which inserts the request, if possible, in the location with the smallest detour. We report the total reward, which is the sum of rewards over the time horizon, and the quantity of the accepted requests. The quantity of accepted requests is not part of the objective function but enables the reader to gain further insight into how the rewards were achieved in addition to the results with regard to the objective function.

The section is divided into the evaluation of the overall performance of the methods, the problem-specific evaluations, which consider the effects of resources and workload allocations on the methods, and the method-specific evaluations, which consider the effects of the number of scenarios on the method. Lastly, we analyse which advantage the degree of freedom of the problem and the decision for the ARS method provide, in order to derive managerial insights for the application of the method.

Table 2
Improvement in total reward and accepted requests compared to CI given different fleet sizes.

Vehicle resources	Rel. reward			Rel. accepted requests		
	ARS	MSA	VFA	ARS	MSA	VFA
Low	**+13.0%**	+7.7%	+9.5%	−1.3%	**+5.8%**	−4.4%
Medium	**+6.5%**	+3.0%	+2.2%	−11.6%	**+1.0%**	−13.1%
High	**+3.2%**	+3.1%	−1.3%	−13.1%	**+2.6%**	−14.0%

Overall performance

In the following, the overall performance of the ARS method and benchmark approaches MSA, VFA and CI are analysed. For ease of reading, the cumulative reward is provided as a relative increase in performance compared to the reward of the CI method. The results are shown as a bar chart in Fig. 5.

The evaluation shows a superior performance of the ARS method compared to the benchmark approaches. In addition, all anticipatory methods show a better performance than the CI method. It is interesting to observe that the ARS method accepts on average 0.7 requests less than the MSA method. Since the ARS method replaces only the consensus method of the MSA with the VFA method, the ARS method has the same scenarios to choose from, learning a better selection than the MSA method. Since the objective function establishes a linear relationship between reward and direct distance transported, the ARS method learns to save resources for requests with further transportation distances. To focus on the amount of accepted requests the service provider may use a different reward function as shown in Appendix C.

Problem-specific performance

In order to compare the performance of the methods depending on problem-specific parameters, experiments are carried out for different fleet sizes. A larger fleet size increases the complexity of the problem and therefore shows the capability of the algorithm to cope with increasing complexity. For a quantity of one, two and three vehicles respectively five, ten and fifteen requests are considered per hour. Table 2 shows the relative improvement in total reward and accepted requests in comparison to the cheapest insertion method. It can be seen that the ARS method provides the best performance but accepts fewer requests at the same time.

Fig. 6 shows the progression of total rewards during the learning phase of the methods for different fleet sizes. The two learning methods are each shown using a solid, ascending line. The MSA as well as the CI method are plotted as a horizontal lines in the graph, as they do

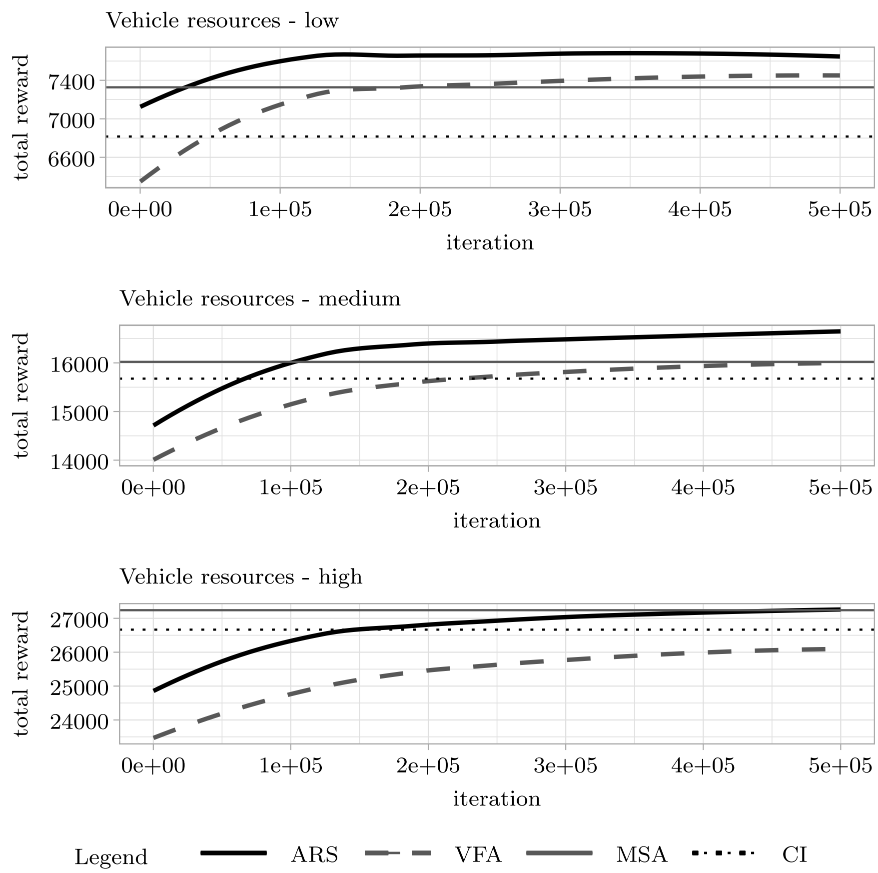

Fig. 6. Simulation results for different fleet sizes.

not contain a learning component, so that the average of the results remains unchanged over time.

It can be observed that the ARS method, shown as solid and black line, achieves higher rewards than the VFA method alone, represented as dashed and grey line. This is presumably due to two effects. Firstly, the integrated method, unlike the individual VFA component, can choose from more than the two options accept and reject, since it has multiple options to choose from in addition to the reject option. Furthermore, the routing of the integrated method includes the sampling method of the MSA approach, which strengthens the anticipation of the stochastic process in routing. It can also be observed that the approximation speed of ARS is similar to VFA even though learning for ARS may be more challenging as more decisions are considered in every state.

Another observation is a higher acceptance rate of the MSA method relative to ARS, VFA and CI. As the integrated method as well as VFA are able to reject requests, the methods seem to save flexibility by accepting fewer requests. The objective function favours requests with long distances which seems to induce the acceptance of fewer and longer requests of the learning methods. In case the providers goal should be maximizing accepted requests in the first place, a different objective function should be chosen which is analysed in Appendix C.

Nevertheless, it can be observed that after a learning phase, the ARS method is able to achieve higher rewards than the MSA method for the considered fleet sizes. The period of the required learning phase to exceed the result of MSA becomes longer with an increasing fleet size. This can be observed in the graphs of Fig. 6 and is attributed to the increasing complexity of the optimization task with a larger fleet size. It is expected that for a fleet size of four or more vehicles, more than 500.000 runs are necessary to achieve a better performance than MSA.

Next, the performance of the methods under different parameters is demonstrated. Three scenarios of request demand are examined. A scenario with low, medium and high workload is assumed with five, ten and fifteen requests per hour. The accumulated rewards as percentage relative to the cheapest insertion method as well as the respective relative amount of accepted requests are reported in Table 3.

Table 3
Improvement in total reward and accepted requests compared to CI given different amounts of incoming requests.

Workload	Rel. reward			Rel. accepted requests		
	ARS	MSA	VFA	ARS	MSA	VFA
Low	−0.4%	**+5.6%**	−8.9%	−3.2%	**+10.7%**	−12.6%
Medium	**+5.8%**	+2.4%	+1.1%	−10.1%	**+2.5%**	−12.1%
High	**+9.0%**	+2.2%	−6.0%	−6.4%	**+4.9%**	−7.9%

Similarly, it can be observed that the ARS method consistently achieves better results than the VFA method, which can be attributed to better anticipation of the random process in routing. Another effect that can be observed is the increase in performance of the ARS method compared to MSA as the demand increases. The fewer requests are received, the more iterations are necessary for the ARS method to return better rewards than MSA. With a request volume of five requests, this point is not even reached. Presumably this circumstance is due to a connection between surplus of requests and room for decision-making. The fewer requests are received, the fewer requests can be rejected to free capacities for future requests as there are simply fewer of them. It seems that if there are five requests, hardly any requests may be rejected. With the ARS method, this still happens less then one request per simulation run, which in the end decreases the performance relative to MSA. The conclusion is that the VFA component of the ARS method is more important when there is more room for decision-making, and the routing component is more important when there is less room for decision-making as an anticipating routing is more important.

Table 4
Improvement in total reward and accepted requests compared to CI over different parameters k.

Parameter k	Rel. reward			Rel. accepted requests		
	ARS	MSA	VFA	ARS	MSA	VFA
$k = 2$	+7.0%	+3.5%	+2.7%	−9.2%	**+3.2%**	−11.1%
$k = 3$	+7.2%	+3.6%	+2.7%	−9.4%	**+3.2%**	−11.1%
$k = 4$	**+7.3%**	+3.6%	+2.7%	−9.4%	**+3.2%**	−11.1%

Method-specific performance

In this section, the influence of method parameters on the performance of the ARS method is examined. In addition, the dependency of the method components on each other is investigated. It can be observed in the experiments with different fleet sizes that one advantage of the ARS method compared to VFA is the larger number of options for the decision. While VFA selects from acceptance or rejection, the ARS method can choose from a number of routing options with an acceptance in addition to rejection. This relationship is examined in more detail in this section. For this purpose, the number of scenarios is varied from two to five.

It can be observed that ARS with 7.3% performance increase relative to CI using three scenarios consistently outperforms VFA with 2.7% relative improvement. After a learning phase, the ARS methods surpass their respective MSA counterparts with the best relative performance of 3.6%. Increasing the amount of used scenarios for the ARS method to four or five scenarios increases the performance negligible, but strongly increases the amount of used computing power therefore runtime. However, decreasing the amount to two scenarios for MSA as well as for the ARS method decreases the performance of the algorithms noticeably. One possible interpretation is that an increase in scenarios is only meaningful if there are rewardful alternative scenarios to consider. This means that if the amount of accepted requests is very small, there may not be many reasonable alternatives to the current routing, so that an increase in the number of scenarios does not necessarily lead to an improvement in performance.

The second element that influences the routing is the k-Opt component which is part of the *integrateRequest* routing method which all algorithms use. To investigate the influence of routing on the service offering component of the ARS method, the influence of the parameter k is examined. The percentage results relative to CI in terms of the objective function as well as the respective relative accepted requests are listed in Table 4. The ARS method, MSA and VFA include the k-Opt component. The number of scenarios was set to two for the simulation.

The superior performance of MSA can be observed in comparison to the cheapest insertion heuristic. However, this is only partly due to the k-Opt method, since the scenario approach is also included. The performance of MSA is not affected by changing the parameter k. This means that the parameter k has no effect on the observed simulation in the long run. Initially a small k slows down the learning progress of the method. Details on the delayed learning progress as well as the learning curve are described in Appendix B.

Dependency of ARS performance to degrees of freedom

The analyses of the experiments suggest that the performance of the ARS method is particularly superior when the degree of freedom of the decision is particularly high. We distinct between two degrees of freedom. First, the degree of freedom of the problem, which we define as the amount of feasible decisions in a state. The more feasible decisions, the higher degree of freedom. One exemplary factor is the number of vehicles on which a request may be allocated. Second, the degree of freedom of the algorithm is described as the amount of decisions the algorithm can choose from. It is reasonable to assume that with a higher degree of freedom, a more efficient route can be chosen so that as few resources as possible are used to obtain a reward and thus more reward may be obtained in the future.

Table 5
Relative transport distance of accepted requests.

Learning phase	ARS	VFA	CI
Start	105.9%	105.1%	100%
End	120%	117.1%	100%

To test this, a measure of efficiency of a decision is first introduced as

$$efficiency = \frac{reward\ gained\ by\ decision}{time\ of\ related\ detour}, \tag{21}$$

in which the detour is measured as the additional **time on route** spent for the decision. The metric efficiency therefore reflects the spent resources to gain reward. To test this assumption, a VFA method that performs a acceptance-or-rejection decision is compared with an ARS method that performs a multiple-scenarios decision (degree of freedom of the algorithm). To reflect the degree of freedom of the problem, the two methods optimize a fleet with one vehicle and a fleet with two vehicles. If the fleet has the size of one vehicle, the complexity of the decision decreases as there is no decision between cars.

Fig. 7 shows the efficiency of the two methods over both fleet sizes. It can be seen that with only one vehicle and thus a low degree of freedom of the problem, there is no efficiency advantage of ARS. We attribute this to the multiple-scenarios decision not yielding an advantage if the problem does not provide sufficient feasible decisions. If the fleet size is increased to two vehicles, the ARS method finds more efficient decisions than the VFA method, making use of the higher degrees of freedom.

Considering the use case, a highly competitive market would lead to a small degree of freedom of the problem, as hardly any requests might be rejected. This market situation would favour the ARS method less. A monopoly of one mobility provider would in turn create room for rejection. This would favour the choice of the ARS algorithm.

Analysis of served customers

Our algorithm changes the service offering for customers. In the following, we analyse how this impacts different types of customers. To this end, we investigate the average trip distance for served customers over all policies.

In order to investigate which aspects the VFA approach learns, it is examined which requests are accepted by the algorithms depending on the corresponding transport distances. This is reasonable, since the reward function rewards the customer transport distance. It will therefore be analysed whether the transport distance of a request has an effect on the decision of the VFA approach. For this purpose, the average length of accepted requests is determined relative to the Cheapest Insertion method at the start and end of the learning process. Since the CI method is taken as the benchmark, it has a value of 100%. The time interval start of the learning phase refers to under 100,000 iterations and end of the learning phase refers to the interval between 450,000 and 500,000 iterations. The result is shown in Table 5.

It can be observed that, as suspected, the VFA-based methods learn to prefer requests with a longer transport distance. The ARS method has an advantage over the VFA method with its relatively simple routing, which is probably due to the fact that the ARS method can better integrate the longer routes into the overall route. It can therefore be deduced that customers with a longer journey have an advantage over customers with shorter requests. In future work it would also be interesting to investigate whether this effect would be reflected in rural regions, where longer journeys are more common.

6. Conclusion

While dynamic vehicle routing problems with stochastic customer requests require a decision to be made on the service offering and

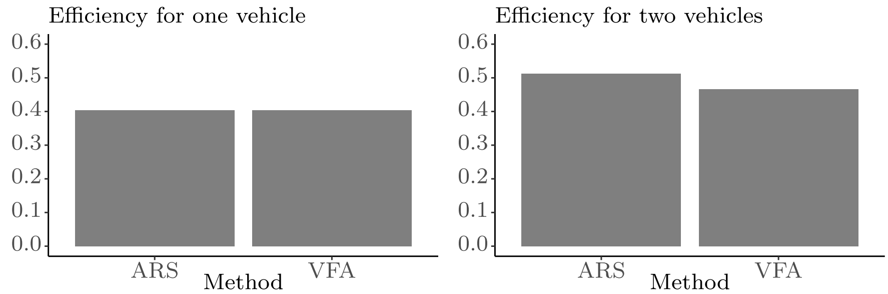

Fig. 7. Decision efficiency for different fleet sizes.

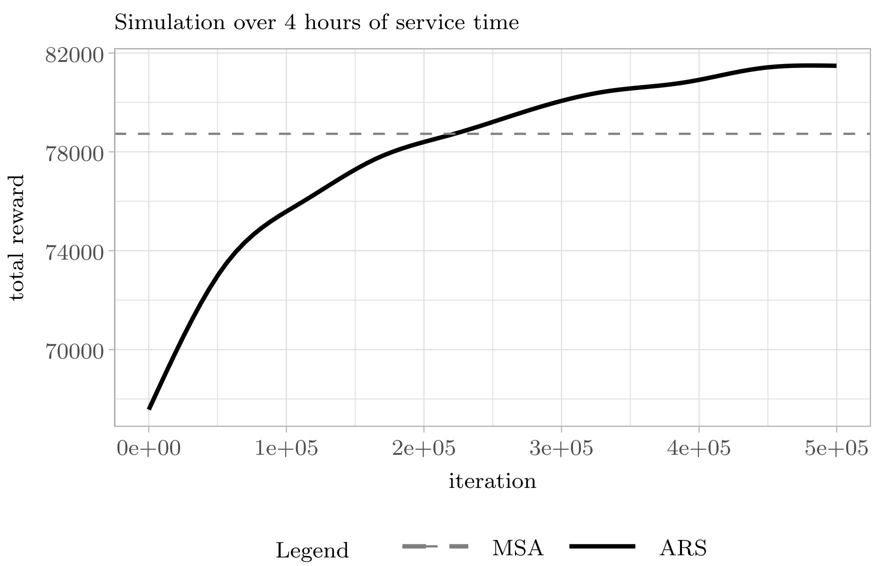

Fig. 8. Simulation extended to a four hour window.

the respective routing, the solutions from the literature typically focus their efforts on one of the two decision components. In this paper, the anticipatory-routing-and-service-offering ARS was proposed, which combines the multiple scenario approach and value function approximation. The ARS method combines an approach that focuses on acceptance decisions of requests with an approach that focuses on anticipatory routing to outperform the single components and even provides significant improvements even in cases where the individual components did not provide much benefit. The influences of the components between each other are analysed. The integration takes place at the consensus function of the MSA, which is replaced by a VFA component.

The performance of the method is tested in experiments using real world data on customer transports of taxis from the New York area. In almost all cases, the integrated method achieves better results than the respective individual components, improving not only the total reward but also the accepted requests. It is found that this performance is particularly high with a heavy workload and thus a large number of decisions to be made. It can be observed that the less room there is for service offering decisions due to fewer requests, the higher the advantage from the anticipatory routing component, as the requests are better integrated into the route. The more requests received, the more important the service offering component becomes. In addition, it can be said that the ARS method has a particular advantage when the problem has a high degree of freedom in decision-making. This means that there are a sufficient number of feasible decisions in a state to choose between. In the future, larger fleet sizes should be tested to confirm the assumption that a longer learning phase may enable ARS to outperform the other approaches. Furthermore, as our methodological combination is general, it might be tested for related SDVR-problems, e.g., from the field of courier services or meal delivery.

Acknowledgements

Marlin Ulmer's work is funded by the DFG Emmy Noether Programme, Germany, project 444657906. We gratefully acknowledge their support.

Appendix A. Longer simulation time window

For the methods discussed in this paper, the simulation effort increases drastically with an increase of the time horizon length. This is presumably due to our implementation of the k-opt method. For this reason, the time window was set to one hour. However, to show that larger time windows still lead to a good performance of the ARS method, Fig. 8 shows the learning curve of the MSA and ARS method. Here, the time window for two vehicles with 50 requests was extended to four hours. The ARS method is still able to outperform the MSA method by approx. 3.6% after 500,000 iterations.

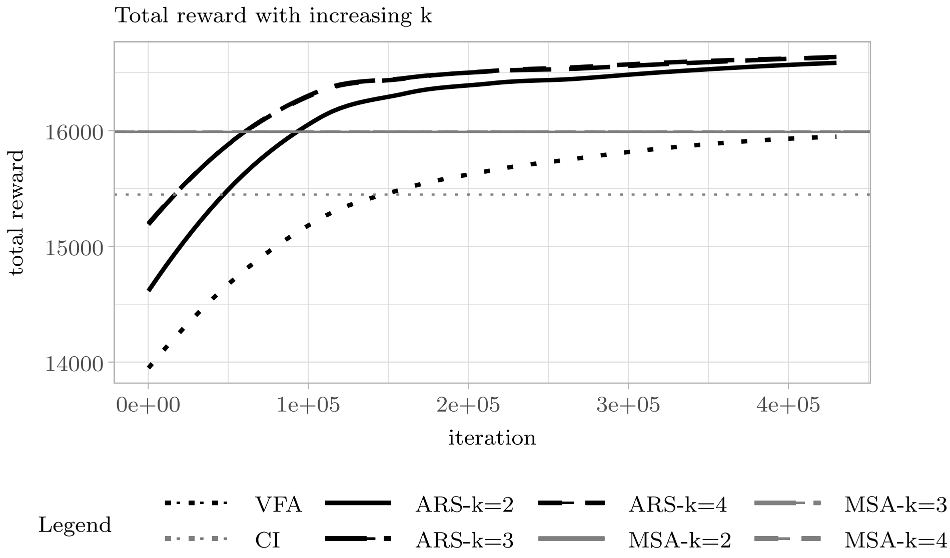

Fig. 9. Simulation results with different k-Opt parameters.

Fig. 10. Simulation with alternative reward function.

Appendix B. Learning progress dependency on the k-parameter

As observed in Section 5, the parameter k for the k-Opt procedure has little effect on performance in the long term. However, it can be observed that a good choice of the parameter early and in the medium term can have a positive effect on the learning progress. When comparing different parameters, it is noticeable that a too small choice of the parameter leads to a delay in the learning progress, as shown by the solid blue line in Fig. 9.

Appendix C. Change of reward function to number of customers transported

To analyse how the ARS method learns under alternative reward models, a new reward function is introduced for this section. In this reward function, each customer pays a fixed price of α for a transport, which makes the price independent of the distance transported. The reward function can be formulated as

$$R(s_k, x_k) = \begin{cases} \alpha & if\ d_k = 1 \\ 0 & if\ d_k = 0. \end{cases} \tag{22}$$

When using the new reward function, the number of accepted requests is maximized. The reward function used previously maximized the distance transported by customers. The results of the experiment can be found in Fig. 10. It can be seen that ARS still shows superior performance. It can also be observed that VFA performs worse than Cheapest Insertion and MSA. However, the gradient of the learning curve of VFA indicates that more learning periods might have been necessary to outperform CI and MSA. It is interesting to observe that the CI method with this reward function leads to good results relative to experiments with the previous reward function.

Appendix D. Learning progress with longer advance notice of customers for their requests

The objective of this section is to investigate how the ARS and the benchmark approaches can deal with less short-term planning. In the experiments of this paper, when announcing a request r_k, the customer can define a pick up time t_k^p in the service horizon T with an arbitrary positive time difference to the request time t_k. A pick up time t_k^p may

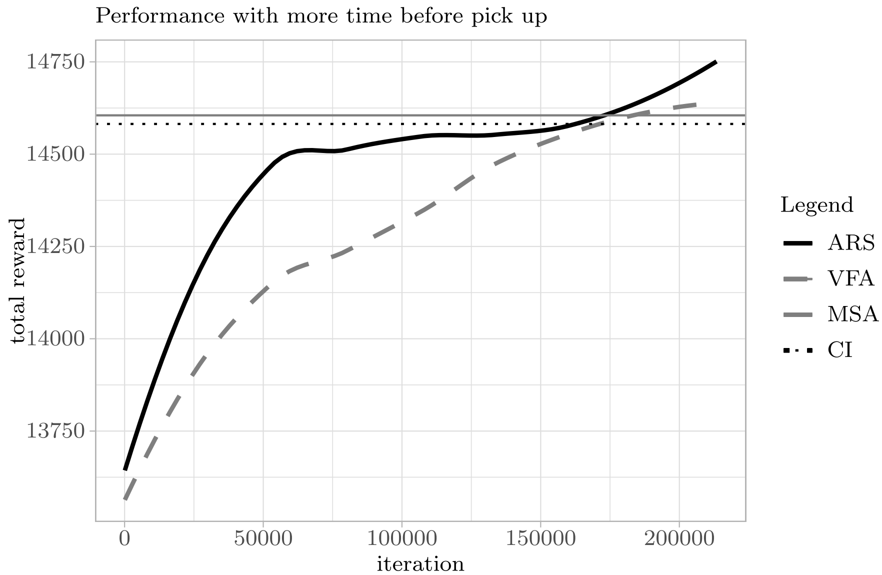

Fig. 11. Simulation with more time between request communication and pick up.

therefore occur at very short notice, which may favour the algorithms' ability to anticipate.

In order to reduce the short-term nature of the planning for the experiment in this section, a fixed interval of 15 min is added to the random positive interval between request r_k and pick up t_k^p. With a time window of one hour, about half of all requests arrive before the start of the service horizon T.

The results of the experiment with less short-term planning can be found in Fig. 11. It can be seen that the ARS algorithms can prevail against the benchmark algorithms. However, it is interesting to see that it does not achieve this as clearly as in the other experiments. This is probably due to the circumstance that the two methods MSA and VFA, which were combined for the ARS method, include strategies of anticipating the stochastic process. The more long-term the planning becomes, the less important these strategies become. This also seems to explain why CI performs so relatively well in this experiments.

References

Agatz, N., Erera, A.L., Savelsbergh, M.W., Wang, X., 2011. Dynamic ride-sharing: A simulation study in metro Atlanta. Procedia-Soc. Behav. Sci. 17, 532–550.

Agussurja, L., Cheng, S.-F., Lau, H.C., 2019. A state aggregation approach for stochastic multiperiod last-mile ride-sharing problems. Transp. Sci. 53 (1), 148–166.

de Armas, J., Melián-Batista, B., 2015. Variable neighborhood search for a dynamic rich vehicle routing problem with time windows. Comput. Ind. Eng. 85, 120–131.

Azi, N., Gendreau, M., Potvin, J.-Y., 2012. A dynamic vehicle routing problem with multiple delivery routes. Ann. Oper. Res. 199 (1), 103–112.

Beaudry, A., Laporte, G., Melo, T., Nickel, S., 2010. Dynamic transportation of patients in hospitals. OR Spectrum 32 (1), 77–107.

Bent, R.W., Van Hentenryck, P., 2004. Scenario-based planning for partially dynamic vehicle routing with stochastic customers. Oper. Res. 52 (6), 977–987.

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

Cortés, C.E., Sáez, D., Núñez, A., Muñoz-Carpintero, D., 2009. Hybrid adaptive predictive control for a dynamic pickup and delivery problem. Transp. Sci. 43 (1), 27–42.

Ehmke, J.F., Campbell, A.M., 2014. Customer acceptance mechanisms for home deliveries in metropolitan areas. European J. Oper. Res. 233 (1), 193–207.

Ferrucci, F., Bock, S., 2014. Real-time control of express pickup and delivery processes in a dynamic environment. Transp. Res. B 63, 1–14.

Ferrucci, F., Bock, S., 2016. Pro-active real-time routing in applications with multiple request patterns. European J. Oper. Res. 253 (2), 356–371.

Forbes, 2020. The future of urban mobility. https://Www.Forbes.Com/Sites/Forbestechcouncil/2020/05/18/the-Future-of-Urban-Mobility/.

Garaix, T., Artigues, C., Feillet, D., Josselin, D., 2011. Optimization of occupancy rate in dial-a-ride problems via linear fractional column generation. Comput. Oper. Res. 38 (10), 1435–1442.

Ghiani, G., Manni, E., Quaranta, A., Triki, C., 2009. Anticipatory algorithms for same-day courier dispatching. Transp. Res. Part E: Logist. Transp. Rev. 45 (1), 96–106.

Häll, C.H., Lundgren, J.T., Voss, S., 2015. Evaluating the performance of a dial-a-ride service using simulation. Public Transp. 7 (2), 139–157.

Hvattum, L.M., Løkketangen, A., Laporte, G., 2007. A branch-and-regret heuristic for stochastic and dynamic vehicle routing problems. Networks: Int. J. 49 (4), 330–340.

Kullman, N.D., Cousineau, M., Goodson, J.C., Mendoza, J.E., 2022. Dynamic ride-hailing with electric vehicles. Transp. Sci. 56 (3), 775–794.

Kuo, R., Wibowo, B., Zulvia, F., 2016. Application of a fuzzy ant colony system to solve the dynamic vehicle routing problem with uncertain service time. Appl. Math. Model. 40 (23–24), 9990–10001.

Laporte, G., Nobert, Y., 1987. Exact algorithms for the vehicle routing problem. In: North-Holland Mathematics Studies, Vol. 132. Elsevier, pp. 147–184.

Maalouf, M., MacKenzie, C.A., Radakrishnan, S., Court, M., 2014. A new fuzzy logic approach to capacitated dynamic dial-a-ride problem. Fuzzy Sets and Systems 255, 30–40.

Marković, N., Nair, R., Schonfeld, P., Miller-Hooks, E., Mohebbi, M., 2015. Optimizing dial-a-ride services in maryland: benefits of computerized routing and scheduling. Transp. Res. C 55, 156–165.

Masson, R., Lehuédé, F., Péton, O., 2014. The dial-a-ride problem with transfers. Comput. Oper. Res. 41, 12–23.

Meisel, S., 2011. Anticipatory Optimization for Dynamic Decision Making, Vol. 51. Springer Science & Business Media.

Mes, M., van der Heijden, M., Schuur, P., 2010. Look-ahead strategies for dynamic pickup and delivery problems. OR Spectrum 32 (2), 395–421.

NYCOpenData, 2018. Yellow taxi trip data. https://Data.Cityofnewyork.Us/Transportation/2018-Yellow-Taxi-Trip-Data/T29m-Gskq.

Pillac, V., Guéret, C., Medaglia, A.-L., 2018. A fast reoptimization approach for the dynamic technician routing and scheduling problem. In: Recent Developments in Metaheuristics. Springer, pp. 347–367.

Pouls, M., Meyer, A., Ahuja, N., 2020. Idle vehicle repositioning for dynamic ride-sharing. In: International Conference on Computational Logistics. Springer, pp. 507–521.

Powell, W.B., 2021. Reinforcement Learning and Stochastic Optimization. John Wiley & Sons.

Pureza, V., Laporte, G., 2008. Waiting and buffering strategies for the dynamic pickup and delivery problem with time windows. INFOR: Inf. Syst. Oper. Res. 46 (3), 165–175.

Riley, C., Van Hentenryck, P., Yuan, E., 2020. Real-time dispatching of large-scale ride-sharing systems: Integrating optimization, machine learning, and model predictive control. arXiv preprint arXiv:2003.10942.

Sarasola, B., Doerner, K.F., Schmid, V., Alba, E., 2016. Variable neighborhood search for the stochastic and dynamic vehicle routing problem. Ann. Oper. Res. 236 (2), 425–461.

Schilde, M., Doerner, K.F., Hartl, R.F., 2014. Integrating stochastic time-dependent travel speed in solution methods for the dynamic dial-a-ride problem. Eur. J. Oper. Res. 238 (1), 18–30.

Shah, S., Lowalekar, M., Varakantham, P., 2020. Neural approximate dynamic programming for on-demand ride-pooling. In: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34. (01), pp. 507–515.

Sheridan, P.K., Gluck, E., Guan, Q., Pickles, T., Balcıog, B., Benhabib, B., et al., 2013. The dynamic nearest neighbor policy for the multi-vehicle pick-up and delivery problem. Transp. Res. Part A: Policy Pract. 49, 178–194.
Srour, F.J., Agatz, N., Oppen, J., 2018. Strategies for handling temporal uncertainty in pickup and delivery problems with time windows. Transp. Sci. 52 (1), 3–19.
Toriello, A., Haskell, W.B., Poremba, M., 2014. A dynamic traveling salesman problem with stochastic arc costs. Oper. Res. 62 (5), 1107–1125.
Ulmer, M.W., Goodson, J.C., Mattfeld, D.C., Hennig, M., 2019a. 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. 100008.
Ulmer, M.W., Mattfeld, D.C., Köster, F., 2018. Budgeting time for dynamic vehicle routing with stochastic customer requests. Transp. Sci. 52 (1), 20–37.
Ulmer, M.W., Thomas, B.W., 2020. Meso-parametric value function approximation for dynamic customer acceptances in delivery routing. European J. Oper. Res. 285 (1), 183–195.
Ulmer, M.W., Thomas, B.W., Mattfeld, D.C., 2019b. Preemptive depot returns for dynamic same-day delivery. EURO J. Transp. Logist. 8 (4), 327–361.
Wu, T., Xu, M., 2022. Modeling and optimization for carsharing services: A literature review. Multimodal Transp. 1 (3), 100028.
Xing, X., Warden, T., Nicolai, T., Herzog, O., 2009. Smize: A spontaneous ride-sharing system for individual urban transit. In: German Conference on Multiagent System Technologies. Springer, pp. 165–176.
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, Z., Liu, M., Lim, A., 2015. A memetic algorithm for the patient transportation problem. Omega 54, 60–71.
Zhang, S., Ohlmann, J.W., Thomas, B.W., 2018. Dynamic orienteering on a network of queues. Transp. Sci. 52 (3), 691–706.