Anticipative Dynamic Slotting for Attended Home Deliveries

Magdalena A. K. Lang¹ · Catherine Cleophas² · Jan Fabian Ehmke³

Received: 15 October 2020 / Accepted: 13 July 2021 / Published online: 6 December 2021
© The Author(s) 2021

Abstract

Attended home delivery requires offering narrow delivery time slots for online booking. Given a fixed fleet of delivery vehicles and uncertainty about the value of potential future customers, retailers have to decide about the offered delivery time slots for each individual order. To this end, dynamic slotting techniques compare the reward from accepting an order to the opportunity cost of not reserving the required delivery capacity for later orders. However, exactly computing this opportunity cost means solving a complex vehicle routing and scheduling problem. In this paper, we propose and evaluate several dynamic slotting approaches that rely on an anticipatory, simulation-based preparation phase ahead of the order horizon to approximate opportunity cost. Our approaches differ in their reliance on outcomes from the preparation phase (anticipation) versus decision making on request arrival (flexibility). For the preparation phase, we create anticipatory schedules by solving the Team Orienteering Problem with Multiple Time Windows. From stochastic demand streams and problem instance characteristics, we apply learning models to flexibly estimate the effort of accepting and delivering an order request. In an extensive computational study, we explore the behavior of the proposed solution approaches. Simulating scenarios of different sizes shows that all approaches require only negligible run times within the order horizon. Finally, an empirical scenario demonstrates the concept of estimating demand model parameters from sales observations and highlights the applicability of the proposed approaches in practice.

Keywords Routing · Revenue management · Attended home delivery · Approximate dynamic programming · Team orienteering problem

✉ Jan Fabian Ehmke
jan.ehmke@univie.ac.at

Catherine Cleophas
cleophas@bwl.uni-kiel.de

¹ RWTH Aachen University, Templergraben 55, Aachen 52062, Germany

² Christian-Albrechts-Universität zu Kiel, Christian-Albrechts-Platz 4, Kiel 24118, Germany

³ University of Vienna, Oskar-Morgenstern-Platz 1, Wien 1010, Austria

1 Introduction

Attended home deliveries (AHD) are both a driver and a result of the seemingly unstoppable growth of e-commerce. When customers have to attend the delivery of products—such as white goods, groceries, or fresh flowers—they expect narrow delivery time slots that fit their personal schedules. For retailers, these expectations come at increasing cost [9, 17], as narrow time slots limit the flexibility of route planning [14]. When retailers use their own fleet of vehicles and drivers or reserve a subfleet of fixed size from a delivery service provider, the variable cost of individual deliveries is negligible. Instead, the largest share of costs in such settings is related to labor and, hence, fixed when labor contracts are strictly regulated as in large parts of Europe. In such a setting, aiming to deliver during each customer's preferred time can cause wasteful idle times, as not all time slots are equally popular. At the same time, a setting where the fixed cost of upkeep for fleet and staff are high calls for delivering as much order value as possible with the given delivery capacity. In consequence, this paper focuses on anticipating the opportunity cost of promising a delivery early within the order horizon and thereby limiting the ability to accept further orders later.

Dynamic slotting helps planners to maintain profitability through limiting of the set of time slots offered to individual customers. Customer choice means that tailoring offers to customers to entice them to select less popular time slots can create a more even distribution of deliveries. Dynamic slotting decisions depend on the current request, the already accepted orders, and orders still expected to arrive in the remainder of the order horizon. They entail solving three connected subproblems: Determining the *feasibility* of delivering the current requested order per time slot, determining the *opportunity cost* of promising the delivery and thereby potentially limiting the resources for accepting future expected orders, and determining the *optimal assortment* of offered time slots to maximise revenue given stochastic customer choice.

However, the underlying complex vehicle routing problem with time windows (VRP-TW) and the uncertainty of stochastic customer choice create significant challenges. Furthermore, dynamic slotting needs to exploit information that is only available on request arrival. As customers expect online retailers to confirm time slots instantly when placing the order, solution approaches need to be computationally efficient to allow rapid responses. Hence, we propose to use a preparation phase before the start of the booking period to support dynamic order acceptance decisions during the booking period.

A primary challenge of anticipative dynamic slotting is the required combination of complex methods from revenue management and vehicle routing. In consequence, related research often only partially succeeds to adhere to respective area's the state of the art. Hence, in this paper, we propose a formalization of the corresponding subproblems and investigate which combinations can be beneficial for anticipative dynamic slotting. Of course, this requires compromises in the interaction of methods from revenue management and vehicle routing, but in the extensive computational study, we highlight beneficial combinations according

to particular demand characteristics and define promising avenues of future research.

For the preparation phase, we create anticipatory schedules by solving the Team Orienteering Problem with Multiple Time Windows (TOP-MTW, [23]) on samples of predicted demand to evaluate the *feasibility* of delivering order requests. Next, we train an offline value function approximation (compare [21]) on simulated demand streams to compute the *opportunity cost* of accepting an order. Lastly, we suggest mechanisms to flexibly fine-tune the *assortment* of offered time slots on request arrival. The resulting approaches rely on preparation to enable quick online decisions and flexible adjustments on request arrival.

In an extensive computational study, we explore the behavior of the proposed solution approaches. Rather than suggesting a one-size-fits-all view of dynamic slotting, the results cause us to propose a differentiated view depending on the specific problem scenario. Simulating scenarios of different sizes demonstrate that all approaches require only negligible run times within the order horizon. Finally, an empirical scenario demonstrates the concept of estimating demand model parameters from sales observations and highlights the applicability of the proposed approaches in practice. We provide access to the code underlying the approaches and study at <https://github.com/SimlabCreator/silful> to support further research in this direction.

2 Literature Review

AHD represent a vivid research field, with contributions focusing on tactical, operational, quantitative, and qualitative aspects. For instance, [10, 20], and [29] survey the topic from different perspectives. Current research on AHD advises to use demand management to profitably assign delivery time slots to customers [13, 18, 28]. Demand management considers characteristics of the current order request, such as basket value, delivery location, and time slot, to evaluate the feasibility and profitability of offering delivery options. Retailers can follow two basic strategies (compare [2]): *Pricing* uses delivery fees to nudge customer choice toward time slots that enable more profitable deliveries. *Slotting* lets retailers control the set of offered time slots. This review focuses on settings that are closely related to the considered dynamic slotting problem, reviewing on scalability, stochastic customer choice, and anticipation. We assume time slot design as in [1] or tactical planning as in [12] to be given.

[5] offer a first approach to tackle dynamic slotting. They combine an insertion heuristic and greedy randomized adaptive search (GRASP) for dynamic route construction during the order period and propose different insertion criteria to maximize overall profit. Later, [4] propose schemes to incentivize customers to choose those time slots from the feasible set that enhance the profitability of delivery. [9] compare mechanisms to maximize the number of accepted orders while ensuring punctual deliveries. These contributions originate from the vehicle routing literature and do not apply sophisticated demand management: They

do not model stochastic customer choice behavior and, except for [5], do not anticipate the profitability of future order requests.

When taking the perspective of revenue management, dynamic slotting can be interpreted as a specific choice-based network problem (compare, e.g., [15, 19]). In particular, it corresponds to the so-called parallel flights problem with availability control as described, e.g., by [30], where time slots represent alternative, substitutable products. [3] solve the dynamic pricing problem as a parallel flights problem by assuming the number of acceptable orders per delivery area and time slot as known. Similarly, [7] approach the slotting problem sequentially by first applying a routing procedure on forecasted order requests to obtain the number of acceptable orders per delivery area and time slot.

However, the dynamic problem differs from static variants in terms of the uncertainty of capacity consumption and left-over capacity: Because the travel time needed per order request depends on the overall set of accepted orders, it can only be known after the final routing. As static approaches do not allow planners to flexibly adjust the estimated capacity during the order period, they impede the potential of fully integrated planning.

Recent contributions in pricing delivery time slots are methodologically related in that they combine vehicle routing and demand management. However, this stream of research primarily decides on the best delivery fee to set per time slot and delivery location. For example, [29] approximate delivery cost based on historical and updated routing schedules. Instead of trying to anticipate implications of not reserving resources for future delivery requests, they consider a delivery-dependent cost as approximated through the increase in travel time when accepting the order. [11] and [28] explicitly anticipate the opportunity cost of assigning delivery resources, but additionally keep the concept of delivery-dependent cost. In contrast, the approaches given here rely on the classical revenue management assumption of irrelevant marginal cost due to a given fleet of vehicles, where only requests that cannot be feasibly delivered with the given fleet would incur an additional travel-time-dependent cost.

[11] present a mixed integer linear program based on comprehensive opportunity cost approximations. However, the computational effort of solving this program impedes its application in practice. In contrast, [28] propose an approximate dynamic programming approach that offers low costs but uses very conservative approximations of delivery cost, inducing higher delivery fees and limiting the deliveries more than necessary. [13] present an approximate dynamic programming approach that uses delivery cost approximations based on current insertion cost in a parallel insertion heuristic, but do not anticipate final delivery cost. More recently, [18] introduced another dynamic slotting approach. They include customer choice modeling and simple approximations of future value and route information, but disregard the possibility of demand segments with differing choice behavior and overlapping time slots. Here, we adapt approaches from [28] and [13] as benchmarks for solution approaches that rely on anticipatory schedule patterns.

Similar to contributions in the area of same-day delivery (for instance, [25, 26]), we rely on approximate dynamic programming to cope with the uncertainty in future demand and decisions. But in contrast to dynamic vehicle routing problems in

same-day delivery, AHD allows sophisticated routing based on all accepted orders before the delivery starts.

3 Problem Statement: The Dynamic Slotting Problem

We consider the following process: Customers fill a shopping basket and provide the delivery address on the retailer's website. In response, the retailer offers a set of delivery time slots. Customers finalize their order by choosing from these slots. The retailer collects orders for a specific delivery period up to a cutoff time, assembles the orders, and organizes their delivery. The model excludes both cancellations and the possibility of amending the time slot after an order has been accepted. We focus on retailers that deliver with their own, dedicated fleet of fixed size, assuming that this renders variable cost negligible. We consider tactical decisions, such as the design of the time slots and the size of the fleet, as input from a previous planning step. Furthermore, we consider time slots for a single delivery period in isolation. Due to scarce travel time, we assume that storage capacity is not a bottleneck and hence do not consider capacity restrictions. To formalize the resulting dynamic problem, we slightly adapt the model of [29], which is the quasi-standard for dynamic pricing.

We consider a delivery region with a single depot. Homogeneous delivery vehicles from a fixed fleet $\mathcal{M}$ start out at the depot and return to it at the end of the delivery period. The delivery period is divided into S (possibly overlapping) time slots, each defined via a start time b_s and an end time q_s .

3.1 Demand Arrival Process and Choice Model

Order requests $j \in \mathcal{J} = \{1, \dots, J\}$ arrive throughout an order period of duration T in a discrete-time process with a homogeneous arrival probability $\lambda \in [0, 1]$ per time step $t = 1, \dots, T$. On arrival, customers announce their actual basket value r_j and choose a time slot s from the tailored *offer set* $\mathcal{S}_j \subseteq \{0, \dots, S\}$. The no-purchase option 0 is always part of the offer set. If the retailer offers no time slots, i.e., $\mathcal{S}_j = \{0\}$, customers abort their order. When customers do select an offered time slot, their order joins the set of accepted orders $\mathcal{O}$.

Every customer j belongs to exactly one demand segment $l_j \in \mathcal{L} = \{1, \dots, L\}$ with probability $\eta_l \leq 1$, $\sum_{l \in \mathcal{L}} \eta_l = 1$. The demand segment determines customers' location, expected basket value r_l , and time slot choice behavior. When dividing the delivery region into a set of disjoint areas, $a \in \mathcal{A} = \{1, \dots, A\}$, the probability that a customer from segment l is located in area a is $\mu_{la} \leq 1$, $\sum_{a \in \mathcal{A}} \mu_{la} = 1$ for all $l \in \mathcal{L}$. An MNL model yields the probability $P_{ls}(\mathcal{S}_{la})$ of a customer from segment l choosing slot s from offer set $\mathcal{S}_{la}$ in area a .

3.2 Optimization Model

The optimization model aims to maximize the expected sum of basket values from accepted orders, given the fixed set of vehicles and expected demand. Per

arriving customer, the best decision depends on further expected orders and the fixed set of accepted orders. The current state $X = [x_{sa}, \dots, x_{sA}]$ is defined as the number of accepted orders x_{sa} per time slot s and area a . Notably, information on the currently considered request could also be considered as part of the current state, as this information is given when deciding on the offer set. However, to clarify the role of choice probabilities in the value function, we follow [29] in considering information on the current request separate from X . Also following that source, we omit the time index in the following for clarity.

To optimize the offer set per customer, dynamic slotting needs to determine the set of feasible time slots, $\mathcal{F}_a(X) \subseteq \{1, \dots, S\}$, as the offer set $\mathcal{S}_{la} \subseteq \mathcal{F}_a(X)$ must only include slots at which a vehicle from the given fleet can feasibly deliver the order to the customer's location. Additionally, accepting the current order at the offered time slot must not make the delivery of any previously accepted orders infeasible. Second, offer set optimization compares the expected opportunity cost of accepting the current customer's request per feasible time slot as opposed to reserving delivery resources for future requests.

To determine the opportunity cost, the following value function describes the maximal expected value for state X from time t until the end of the order period:

$$\begin{aligned} V_t(X) &= \max_S \lambda \sum_{l \in L, a \in A} \eta_l \mu_{la} \sum_{s \in \mathcal{S}_{la} \subseteq \mathcal{F}_a(X)} P_{ls}(S_{la}) [r_l + V_{t+1}(X + \mathbb{1}_{as})] \\ &\quad + [1 - \lambda \sum_{l \in L, a \in A} \eta_l \mu_{la} \sum_{s \in \mathcal{S}_{la} \subseteq \mathcal{F}_a(X)} P_{ls}(S_{la})] V_{t+1}(X) \\ &= \lambda \max_S \sum_{l \in L, a \in A} \eta_l \mu_{la} \sum_{s \in \mathcal{S}_{la} \subseteq \mathcal{F}_a(X)} P_{ls}(S_{la}) [r_l - (V_{t+1}(X) - V_{t+1}(X + \mathbb{1}_{as}))] \\ &\quad + V_{t+1}(X), \forall X \in \mathcal{X} \end{aligned} \quad (1)$$

In Equation (1), the unit vector $\mathbb{1}_{as}$ indicates an additional accepted order for area a and time slot s . Note that the no-purchase option is not part of the set of feasible time slots $\mathcal{F}_a(X)$. Therefore, the no-purchase option is covered by the second part of the equation. After the cutoff time T , no delivery-dependent cost arises, as only those requests that could be delivered with the given fleet were accepted. However, also, no additional value can be collected:

$$V_{T+1}(X) = 0, \forall X \in \mathcal{X} \quad (2)$$

Thus, a state's value is recursively defined via the possible next states. In every time step, the number of accepted orders either remains stable, when there is no arrival or the customer chooses the no-purchase option, or it increases, when the customer chooses an offered time slot.

We define the opportunity cost for time slot s and current customer j from area a as $\delta_{x_{tas}} = V_{t+1}(X) - V_{t+1}(X + \mathbb{1}_{as})$. The optimal offer set depends on the time slot choice probabilities predicted for the customer segment. Thus, per request j from area a arriving at time t , the following policy determines the offer set for state x :

$$\mathcal{S}_j^* = \arg \max_{\mathcal{S}_j} \sum_{s \in \mathcal{S}_j \subseteq \mathcal{F}_a(X)} P_{l_j s}(\mathcal{S}_j) [r_j - \delta_{xtas}]. \quad (3)$$

Given the opportunity cost and the set of feasible time slots, the problem of assortment optimization could be efficiently solved through revenue-ordered sets or via a standard simplex solver (compare [8]). Nevertheless, the computing opportunity cost requires to solve the DP and the routing problem, which is also complex. Moreover, checking feasibility requires to solve an NP-hard VRP-TW [22]. Thus motivated, the following section presents solution components that enable computationally tractable approaches.

4 Solution Components for Anticipative Dynamic Slotting

In this section, we introduce solution components applicable to the sub-problems of *determining the feasibility* of delivering an order, *computing the opportunity cost* of order acceptance, and *optimizing the assortment* per customer. As shown in Fig. 1, we differentiate a simulation-based preparation phase (anticipation) ahead of the order period and dynamic decision making within the order period (flexibility):

- We consider two alternative components to check for feasibility: *anticipative schedule patterns* and *ad hoc routing*.
- We propose to anticipate opportunity cost by preparing an *off-line value function approximation model (VFAM)*. This model can be trained on simulated demand samples during the preparation phase and supplies approximated opportunity cost per arriving customer within the order period.
- Given feasibility and opportunity cost, dynamic slotting needs to optimize the assortment of offered time slots. As by its nature, any anticipation can be flawed, we propose to complement optimization results via *theft-based mechanisms* to enable flexible decisions.

Fig. 1 Solution components in preparation phase and online policy

4.1 Determining the Feasibility of Deliveries

In Section 4.1.1, we describe how to compute *anticipative delivery schedules* ahead of the order period and aggregate them into *schedule patterns*. We evaluate the feasibility of accepting an order for a time slot based on their fit with the anticipative patterns within the order period. In Section 4.1.2, we consider *ad hoc routing* to check the feasibility of the request in the light of already accepted orders within the order period.

4.1.1 Anticipatory schedule patterns

To anticipate schedules even before actually accepting orders, we solve the team-orienting problem with time windows (TOP-MTW) on demand samples. We draw these samples from stochastic distributions of basket values, time slot preferences, locations, and arrival times given by estimated demand model parameters. Solving the TOP-MTW yields the most valuable set of order requests to accept while ensuring feasible deliveries; compare Appendix 10 for the related mathematical model. Solving the TOP-MTW on multiple demand samples builds a pool of anticipatory delivery schedules that captures the inherent stochasticity of demand.

For acceptable computational effort, we implement the GRILS heuristic by [23]. This heuristic only requires a few parameters and thereby limits the need for parameter tuning. Moreover, it is fast enough to solve for multiple demand samples in acceptable time. GRILS is an iterated local search algorithm, hybridized with GRASP—refer to [23] and [27] for more details.

To support decision making on request arrival, we aggregate the resulting delivery schedules into anticipatory patterns (compare Fig. 2). We divide the delivery region into a set of disjoint delivery areas. The pattern details the schedule's number of accepted orders per delivery area and time slot, abstracting from individual orders. Boolean patterns indicate only whether or not a delivery is planned per area and time slot.

Dynamic slotting can use anticipatory schedule patterns to evaluate *feasibility* during the booking period. Feasibility is given when a pattern includes deliveries to the customer's area during the considered time slot. This is true if any delivery

Fig. 2 Transforming a delivery schedule into an anticipatory patterns

is planned in a Boolean pattern or if the number of deliveries in a regular pattern exceeds the number of already accepted orders for the combination of area and time slot.

Anticipatory schedule patterns also enable a more gradual check to quantify the *favorability* of offering time slots. To that end, we evaluate the fit of accepted orders with the Boolean schedule patterns. The favorability check computes a dissimilarity measure to quantify the difference between the accepted orders and each anticipatory schedule pattern. The higher the dissimilarity measure, the lower the fit. Notably, any measure that can quantify the distance between two matrices can apply here. The choice of dissimilarity measure certainly affects outcomes, e.g., when weighing particular parts of patterns differently to emphasize rush hours or dense settlements. We rely on a simple absolute distance here and point out the related opportunity for research in the conclusion.

Aggregating the dissimilarity measures over multiple patterns quantifies how favorable it would be to accept a request per feasible time slot. Any measure that supports direct comparisons can measure dissimilarity. For details on Boolean dissimilarity measures, compare Appendix 8.

Furthermore, various ways of aggregating dissimilarity measures over multiple schedule patterns are conceivable. Given stochastic problem characteristics such as customer locations and time slot preferences, schedule patterns are heterogeneous even when created for the same problem setting. Not all patterns are equally relevant to the spatial distribution of a demand sample, especially towards the end of the order horizon. Looking at the smallest dissimilarity measure means only consulting the pattern that best fits the current sample.

We let a threshold parameter ϕ^{max} define the maximum increase in aggregated dissimilarity that a request can cause to be considered favorable for acceptance. Fig. 3 provides an example: For the first two samples, accepting the request causes the dissimilarity measure for pattern 1 to increase to d^n . Thus, the aggregated dissimilarity measure is d^n and $\phi = d^n$. If $\phi^{max} = 0$, the last order could not be accepted and the smallest dissimilarity measure across schedule patterns will never exceed 0, and thus the set of accepted orders has to perfectly fit at least one pattern. When

Fig. 3 Exemplary (Aggregated) Dissimilarity Measures

setting $\phi^{max} = d^n$, a new order can also be acceptable if it is in a neighboring combination. Increasing ϕ^{max} over the order horizon increases the flexibility of assigning left-over delivery resources to the actual requests.

4.1.2 Ad Hoc Routing

Alternatively, dynamic slotting can check the feasibility of delivering an order in a time slot through ad hoc routing within the order period. Such a routing finds the best feasible insertion position in the current schedule based on the time required to serve the customer. This is the sum of the service time and the additional travel time caused by the new order. A time slot is feasible if at least one feasible insertion position exists. In contrast to anticipatory schedule patterns, ad hoc routing does not account for future expected order requests and does not prioritize valuable customers.

4.2 Anticipating Opportunity Cost Through Value Function Approximation Models

We propose to use the preparation phase to train an off-line VFAM that approximates the opportunity cost $\hat{\delta}_{t\sigma(\mathcal{O}_t)s} = \hat{V}_t(\sigma(\mathcal{O}_t)) - \hat{V}_t(\sigma(\mathcal{O}_t \cup o_s))$ per time slot s . Here, function $\sigma(\mathcal{O}_t)$ extracts features from the set of accepted orders. Specifically, we train the VFAM $\hat{V}_t(\mathcal{O}_t)$ from simulated arrival streams by minimizing the squared loss function:

$$L(\mathcal{O}_t) = \frac{1}{2} \mathbb{E}[(\hat{V}_t(\mathcal{O}_t) - \bar{V}_t(\sigma(\mathcal{O}_t)))^2]. \quad (4)$$

In this, $V_t(\mathcal{O})$ denotes the observed value.

We apply a Monte Carlo approximate dynamic programming approach to train the model, considering each simulated order period as one episode. In particular, we implement generalized policy iteration by alternating between policy evaluation and improvement on an episode-by-episode basis (see [24]). Fig. 4 illustrates the approach and its two main steps.

After simulating one episode, we update the model for policy improvement. Over time steps $0 \leq t \leq T$, the observed future return $V_t(\mathcal{O}_t)$ serves as the target value for a stochastic gradient descent step toward minimization of the objective function:

$$V_t(\mathcal{O}_t) = \sum_{k=t+1}^T R_k(\mathcal{O}_k) \quad (5)$$

$R_t(\mathcal{O}_t)$ is the observed reward in time step t , defined by either 0—in case of no arrival or the customer choosing the no-purchase option—or the respective basket value

Fig. 4 Generalized Policy Iteration Approach

of an accepted order. Equation (5) does not include a discounting factor because in this setting, future rewards are not less valuable than immediate rewards. Thus, the approach is far-sighted.

Shuffling the state-value pairs before stochastic gradient descent updates decreases the correlation between subsequent data points. Per state-value pair, we perform one stochastic gradient descent step. A learning rate α defines the step size and slowly decreases over the training phase via annealing, that is $\alpha^i = \alpha/(1 + i/Q)$ with Q as experimental parameter. Moreover, we introduce a momentum to smooth out variations from individual gradient descent steps by considering previous updates. The momentum weight ω has to be calibrated via preliminary experiments.

We alternate policy evaluation and improvement over training samples. The last state of the approximation model serves as model during the actual order period.

4.3 Assortment Optimization Throughout the Order Period

Based on opportunity cost $\hat{\delta}_{t\sigma(O_i)s}$, assortment optimization computes the offer set $\hat{S}_j$ per arriving order request j given some candidate set of feasible or even favorable time slots $\mathcal{C}$:

$$\hat{S}_j = \arg \max_{\hat{S}_j \subseteq \mathcal{C}} \sum_{s \in \hat{S}_j} P_{I,s}(\hat{S}_j)[r_j - \hat{\delta}_{t\sigma(O_i)s}]. \quad (6)$$

For an MNL model of customer choice, the assortment optimization problem can be efficiently solved by the concept of revenue-ordered sets (refer to, for instance, [8]). Time slots are sorted by decreasing value-add (basket value—opportunity cost) and consecutively included in the offer set for as long as the expected value increases.

4.4 Adapting the Assortment Through Theft-based Mechanisms on Customer Arrival

As assortment optimization relies on anticipated demand and delivery schedules, it may not perfectly match the arriving order requests. Therefore, we propose additional *theft-based mechanisms* to flexibly adjust the assortment on request arrival. We specify three mechanisms that allow order requests to “steal” delivery capacity from a neighboring area of the anticipatory schedule pattern. For illustration, we consider a delivery area with three neighbors (see Fig. 5), where a time slot is not feasible according to previous checks.

Fig. 5 Theft-based flexible mechanisms

Theft Mechanism 1 In Mechanism 1, there is at least one accepted order for the currently requested delivery area and time slot, such that delivering another order is less likely. Therefore, all neighboring areas that still have capacity for the respective time slot are candidates for theft. Per relevant pattern, the final theft candidate is the area with the lowest opportunity cost for the respective time slot.

Theft Mechanism 2 In Mechanism 2, no previous order has yet been accepted in the current delivery area for the time slot yet but feasibility checks allow for two more orders to be accepted in neighboring areas. A theft is only allowed if this number of feasible deliveries exceeds the neighbors' expected demand for the time slot. In this case, assigning the time slot entails stealing two units of capacity from neighbors. The opportunity cost are computed accordingly. Such a theft can be especially valuable for time slots with very low popularity.

Theft Mechanism 3 In Mechanism 3, deliveries to a neighbor area are still feasible and there was already an order accepted for a neighbor time slot in the current area. A neighbor time slot either ends with the beginning of the current time slot, begins with the end of the current time slot, or overlaps with the time slot. Such a theft means that, as the vehicle already visits the area for a neighbor time slot, it can leave the area later or visit the area earlier to serve a customer in the current time slot.

5 Solution Approaches

We assemble three dynamic slotting approaches from the components introduced in Section 4. Table 1 lists salient differences in terms of how the alternative approaches employ anticipative and flexible components. All approaches follow the outline given by Fig. 1, exploiting a preparation phase to prepare the online policy. They differ in their reliance on anticipatory patterns and in the optional implementation of checks for favorability and theft-based mechanisms. Notably, all suggested approaches use the preparation phase to train a VFAM for computing opportunity cost. However, as detailed in this section, the related VFAM are individually specified to fit well with the overall approach. When implemented in an appropriately modular fashion, these solution components allow for a combinatorial array of variants that by far exceeds the scope of a single paper.

Accordingly, this section and the computational study focus on a selection of promising options. In the further text, we consider two approaches from existing references, benchmarking ideas from [13] and [28]. However, we significantly adapted these approaches to make the resulting approaches comparable and consistent to the idea of focusing on opportunity cost rather than variable delivery cost and to the concept of using anticipatory patterns. Further variants and parameterizations can be tested using the code repository that accompanies this paper.

5.1 Routing-based-based Approach (rout)

As a benchmark from the literature, adapted from [13], we implement a routing-based dynamic slotting approach that considers remaining travel time (rout-IC).

Table 1 Dynamic Slotting Variants

Approach	Explanation	Anticipation	Flexibility
routing-based: ad hoc routing + linear VFAM			
route-IC	VFAM features remaining travel time, adapted from [13]	+	+++
routing-and-pattern-based: ad hoc + anticipatory patterns + linear VFAM			
rpat-d0-#	strict favorability check, VFAM features only the number of accepted orders, adapted from [28]	++	+
rpat-d0-EC	strict favorability check, VFAM additionally features anticipated travel time	++	+
rpat-d2-EC	relaxed favorability check, VFAM additionally features anticipated travel time	++	++
rpat-d2-EC-TW	like rpat-d2-EC but decomposes anticipatory schedule patterns per time slot	++	++
rpat-d2-EC	relaxed favorability check, VFAM additionally features remaining travel time	++	++
dependent-on-patterns: checks feasibility and favorability on patterns; nonlinear VFAM			
pat	without flexible theft mechanisms	+++	+
pat-th ⁺	theft mechanisms support flexible decisions	+++	++

This approach decides the feasibility of accepting requests based on ad hoc routing, but is adapted from the original to neglect variable cost of delivery. Checking the feasibility based on ad hoc routing provides a candidate set of time slots $\mathcal{C}$. To compute the opportunity cost $\hat{\delta}_{i\sigma(\mathcal{O})_s}$ of assigning time slot $s \in \mathcal{C}$, rout relies on a VFAM with a linear combination of features, including an estimate of the remaining travel time.

The VFAM features describe the current state as follows. The remaining time in the order period is approximated by $\iota = T + 1 - t$. The numbers of already accepted orders per time slot, x_s , indicate the consumption of capacity. The required amount of travel time per customer depends on their location and their fit to delivery schedules for a given time slot. Therefore, rout-IC additionally considers the remaining travel time, $d \in \mathbb{R}$. This can be easily extracted from delivery schedules computed for the feasibility check. Equation (7) defines rout's VFAM with $\beta_0 \in \mathbb{R}$ as intercept and $\beta_g \in \mathbb{R}$ as coefficient for feature g . Note that all feature values ι and x_s are normalized between 0 and 1, indicated by, for instance, $\check{x}_s$. For normalization, the maximum value for ι is the order horizon length T ; the largest value for x_s is the maximum number of accepted orders in an anticipatory pattern for time slot s across the pool of patterns.

$$\hat{V}_t^{rout}(X, d, \iota) = \beta_0 + \sum_{s \in \mathcal{C}} \beta_s \check{x}_s + \beta_d \check{d} + \beta_\iota \check{\iota} + \beta_{x\iota} (1 - \check{\iota}) \left(\sum_{s \in \mathcal{C}} \check{x}_s \right) \quad (7)$$

The interaction term $\beta_{x\iota} (1 - \check{\iota}) (\sum_{s \in \mathcal{C}} \check{x}_s)$ between elapsed time in the order horizon and overall number of accepted orders allows opportunity cost to depend on the time in the order horizon and, thus, remaining demand. In other words, it serves to counteract the negative effect of accepting another order for time slot s . This effect is more pronounced toward the end of the order horizon, when there remains less room for negative effects from accepting orders rather than reserving capacity for future orders.

$$\hat{\delta}_{Xdis}^{rout} = -\beta_s + \beta_d \nabla_{ds} - \beta_{x\iota} (1 - \check{\iota}) \quad (8)$$

The resulting opportunity cost is defined as given by Equation (8), with ∇_{ds} as the additional travel time caused by accepting the new order for time slot s . Thus, the linear model allows opportunity cost that depend both on time and remaining capacity.

5.2 Routing-and-pattern-based Approach (rpat)

The second approach, *routing-and-pattern-based* (rpat), checks feasibility via ad hoc routing and further reduces the candidate set by evaluating the favorability of slots. To this end, it compares the resulting delivery schedule to a pool of anticipatory schedule patterns generated as described in Section 4.1.1. rpat also approximates opportunity cost via a linear VFAM before optimizing the assortment. rpat-d0-... and rpat-d2-... variants differ in how strictly they apply the favorability check (compare Table 1): rpat-d0-... variants only accept orders that fit with an anticipatory schedule pattern ($d_n = d_o = 2 > 0$ and $\phi^{max} = 0 \rightarrow d0$). As a more flexible

option, rpat-d2-... variants accept orders from neighbor areas in the last third of the order period ($d^n = 2$ and $d^o = 4$ and switch from $\phi^{max} = 0$ to $\phi^{max} = 2 \rightarrow d2$). Note that the choice of d^n and d^o here is arbitrary, invariant to scaling, and reflects a proportion rather than an absolute bound. Additionally, we differentiate variants by VFAM as follows:

- We implement rpat-d0-# as a benchmark from the literature, adapting [28], where the number of feasible deliveries is anticipated from existing schedules. In contrast to the original approach by [28], this variant considers a set of given, disjoint areas and relies on anticipatory schedule patterns rather than specific routings for approximating opportunity cost.
- As ad hoc routes do not anticipate final routing schedules, rpat-d0-EC, rpat-d2-EC, and rpat-d0-EC-TW additionally use anticipated travel times obtained from anticipatory schedule patterns to describe the current state of assigned resources.
- rpat-d0-EC-TW increases flexibility by decomposing patterns per time slots. However, this also breaks down relationships between subsequent time slots and diminishes the guiding role of the anticipatory patterns.
- rpat-d2-IC considers the remaining travel time, rather than the overall travel time, as a feature.

5.3 Pattern-based Approach (pat)

The fully anticipatory dependent-on-patterns approach (pat) evaluates both the feasibility and the favorability of time slots based on anticipatory schedule patterns. On request arrival, pat approximates opportunity cost from the accepted pattern X , a pool of relevant patterns, and the VFAM.

Notably, we considered some variants of pat in experiments that are not detailed in this paper for parsimony. The results lead us to neglect evaluating these variants in further detail here. Specifically, we combined the nonlinear value function and a test for ad hoc feasibility as described for rout. However, the nonlinear value function is trained on area-level and assumes capacity from the patterns. The ad hoc check would not fit to expected capacities and there is a drift, leading to poor results. As another variant, we ran preliminary experiments using a linear value function instead of a nonlinear value function. The nonlinear value function outperformed the alternative, causing us to recommend it for this approach.

Anticipatory Pattern Pool After each order acceptance, pat updates the pool of relevant patterns. All anticipatory schedule patterns that still perfectly match the accepted orders are relevant. When no pattern remains relevant after the assignment, the time slot is not feasible and does not enter offer set optimization.

Approximating Opportunity Cost pat determines opportunity cost by first individually computing VFAs for all relevant patterns and subsequently aggregating the results as follows: First, the VFAM approximates the value $V_{t+1}^{pattern}(X)$ of not assigning any time slot to the current request given the capacity defined by the pattern.

The respective optimization problem is a variant of a classical revenue management problem with substitute products and multiple demand segments as detailed in Appendix 11.2. Second, the VFAM computes the remaining value for an additional assignment per feasible time slot, i.e. $V_{t+1}^{pattern}(X + \mathbb{1}_{as})$ for all time slots where the number of deliveries in the anticipatory pattern still exceeds the number of accepted orders. Because the opportunity cost approximations are aggregated over patterns, the expected value from not assigning any time slot is the average of all $V_{t+1}^{pattern}(X)$. The expected value for assigning a time slot is the average of all $V_{t+1}^{pattern}(X + \mathbb{1}_{as})$ per still relevant pattern after assignment.

Non-linear VFAM Due to its individual consideration of anticipatory schedule patterns, the VFAM accounts for the demand arrival rate and mixtures of segments varying across delivery areas. The area-specific demand segment proportions affect the opportunity cost of time slots through the segments' heterogeneous basket value distributions and choice models. We propose to implement a holistic VFAM that can capture the differences over areas. The model features describe the area-specific state and capture the inherent complexity via non-linear relationships. Fortunately, the same number of simulated order periods provides more training data for such an approach compared to rout and rpat, as it supplies not only one observation per time step but one observation per time step and delivery area.

Artificial Neural Network (ANN) To model non-linear dependencies, we implement a feed-forward ANN with a single hidden layer to approximate the value function. We calibrate the appropriate number of hidden nodes via preliminary experiments. For features, the remaining capacity per time slot is an intuitive choice. We operationalize this through the capacity according to an anticipatory pattern minus the number of accepted orders per combination. In contrast to rout and rpat, which represent future demand via the remaining time in the order period, pat uses the expected arrivals per demand segment. As discussed earlier, the proportions of the demand segments per delivery area determine the value of future demand. Thus, these features enable the model to capture the demand structure per delivery area.

VFAM Learning The learning phase conforms with the general approach introduced in Section 4.2 with one exception: The policy applied during policy evaluation does not match the online policy. As pat's VFAM approximates future expected value for a specific anticipatory pattern, the online policy aggregates these values only after calculating them for all relevant patterns. During learning, a randomly drawn pattern determines the capacity for that specific order period simulation. We update the ANN weights over time steps $0 \leq t \leq T$ and per delivery area via stochastic gradient descent steps and backpropagation.

Theft-Based Mechanisms Relying on reserving capacity to delivery areas causes pat to be less flexible in reacting to actual demand. Therefore, we propose to combine pat with the flexible theft-based mechanisms introduced in Section 4.4, creating the variant pat-th⁺.

6 Computational Study

This computational study uses synthetic demand settings to represent a variety of real-world situations as differentiated by the location and value distribution of demand. We analyze specific situations in detail to trace the implications of trading off anticipatory planning and flexible decision making. On large-scale scenarios, we particularly evaluate the required computational effort. Lastly, we present results from a demand setting derived from empirical e-grocer data to evaluate real-world applicability. The results emphasize the value of anticipative planning, but also highlight its dependence on characteristics of the problem setting. Full access to the code, solution approaches and variants, and to the scenario data is available at <https://github.com/SimlabCreator/silful>.

6.1 Computational Setup

The computational study relies on a discrete-event-based, stochastic simulation (compare [16]). To account for stochastic effects, each simulation experiment applies one approach to a specific problem scenario over 100 order periods. For each experiment, we perfectly replicate the set of stochastic order requests from the problem scenario to evaluate approaches under *ceteris paribus* conditions. The offline trainings of the VFAM each use 5,000 demand samples. Preliminary experiments yielded a learning rate α of 0.0001, a momentum of 0.9, and an annealing value of 4,000 as a good parametrization across approaches and problem settings. To determine travel times between customers, we use the Euclidean distance multiplied by 1.5, and assume an average vehicle speed of 30km/h. We run simulations on an Intel Xeon CPU E5-1650 v4 with six 3.6GHz cores und 32GB RAM. The average run-time per order request arrival is below 1 millisecond for all approaches and settings. Section 6.3 details the computational effort of the preparation phase as dependent on the scale of the instance.

Benchmarks We compare all dynamic slotting variants in Table 1. We scale all results on a first-come-first-serve policy (FCFS), which computes the feasibility of accepting a request based on ad hoc routing and offers every feasible time slot. We use rout-IC to benchmark effects from relying on anticipation in checking the feasibility and value of orders. We compare several variants of rpat to evaluate effects of VFAM feature selection (rpat-d0-# vs. rpat-d0-EC# vs. rpat and rpat-d2-EC vs. rpat-d2-IC), favorability checks (rpat-d0-EC vs rpat-d2-EC), and decomposing anticipatory patterns per time slot (rpat-d2-EC vs rpat-d2-EC-TW). Last but not least, two pat variants (pat vs. pat-th+) benchmark the effect of fully relying on anticipatory patterns and of flexible theft-based mechanisms. For all approaches, a final routing based on adaptive local neighborhood search checks feasibility of the accepted orders per simulation run.

Pattern Generation We generate anticipatory schedule patterns via the GRILS heuristic, setting the maximum number of iterations without improvement to 60

and the greediness factor to 1. For pat, we remove selected orders from TOP schedules to be prepared for unexpectedly large travel times within delivery areas: when orders are located far apart within their assigned delivery area but the respective TOP schedule incorporated orders close to each other, the risk of infeasible orders occurs due to the aggregation. Thus, in case the slack of a time slot is smaller than the travel time required for the diagonal of an area, an order is removed to reduce that risk.

Measuring Performance All dynamic slotting approaches aim to maximize the *basket value* collected over the order period. When a final routing determines infeasible accepted orders, we adjust the basket value to model the cost of denied services as common in the overbooking literature [6]. The first infeasible order reduces the collected basket value by the setting's mean basket value $\hat{r}$. Any additional infeasible order incurs cost that is 110% of the prior cost. Thus, the n^{th} infeasible order causes a punishment value of $1.1^{n-1}\hat{r}$.

We measure additional indicators to analyze effects of problem setting and slotting variant. For instance, the *number* of accepted orders can indicate whether a higher basket value indicator comes from higher individual basket values or from a higher number of accepted orders. *Idle times* in delivery schedules indicate logistics performance and, similar to the highest basket value not collected, help to identify regret in terms of left value.

Indicators are averaged across simulation runs and are scaled in relation to the FCFS outcome. Thus, the value is divided by the average FCFS value of that problem setting. For instance, a basket value of 1.12 indicates a 12% higher basket value than what was earned by FCFS.

6.2 Synthetic Scenarios

Each problem scenario defines a specific setting of the following characteristics: set of time slots, region size, number of vehicles in the fleet, and demand setting. Every order period includes 500 time slices ($T = 500$). Delivery regions are divided into a grid of 36 (6x6) delivery areas. All synthetic scenarios consider a 12-hour delivery period.

Table 2 lists and motivates all further characteristics. The demand setting is defined by a mix of *demand segments* and the *arrival rate of customer requests*. Each segment shares a *time slot choice model*, *basket value distribution*, and *location distribution*. Different combinations of segments represent different real-world situations. Each setting features one of four combinations of two demand segments (see Table 2). In each combination, Segment 1 obtains a weight $\eta_1 = 0.75$, while Segment 2's weight is $\eta_2 = 0.25$. We conduct simulation experiments on 3(time slot sets)x4(demand segment combinations)x6(region sizes, vehicles/arrival rates)=72 synthetic problem scenarios.

Table 2 Computational study: Settings

Problem settings	
Characteristic	Settings
Set of time slots	Shorter time slots allow less rearrangement of orders in routes and further limit possible distances between locations within same time slots. Thus, altering the length of time slots enables to analyse algorithm's performance on different degrees of flexibility in route planning. We divide the delivery period into three alternative sets of time slots, terming a set of 12 disjoint one-hour time slots ts_1h , a set with six disjoint two-hour slots ts_2h ; and the set of overlapping two-hour time slots $ts_overlap$.
Region size	Scenarios feature one of two region sizes, a larger region of 15x15km and a smaller region of 10x10km. In all scenarios, the depot is located on the right upper side of the region (see Fig. 6).
Number of line vehicles	Small region settings can include two or three vehicles, whereas scenarios in the larger region always feature three vehicles. This generates reasonable proportions between demand and available vehicles, slightly varying the scarcity of resources by altering region size, number of vehicles, and number of requests.
Time slot choice models	Per set of time slots, we define customer choice behavior by time slot utility values (for details, see Appendix 12). Choice model F represents flexible customers, who consider all time slots but prefer the late afternoon. Customers that exhibit behavior I are rather inflexible and consider only a subset of time slots, especially in the morning and late afternoon.
Basket value distributions	Demand segments' basket values come from one of two alternative log-normal distributions. Distribution H has a high mean of 75 with distribution parameters $\mu_1 = 4.317$ and $\sigma_1 = 0.1$. Distribution L creates low basket values with an average of 50 and parameters $\mu_2 = 3.912$ and $\sigma_2 = 0.15$. Basket values are multiplied with 0.3 to obtain profits, as suggested in [29].
Location distributions	Customer locations can be distributed uniformly over the region, in two clusters (a or b) or four clusters. Figure 6 visualises this, indicating the areas of the region that include customer locations in grey.
Arrival rates	For two vehicles, arrival rates are 0.3 and 0.27 (-10%). For three vehicles, arrival rates are 0.45 or 0.405.
Demand segment combinations	
Combination	Description
<i>Center-uniform</i>	City center with uniform location distributions: Segment 1 features the flexible time slot choice behavior (F), low basket values (L), and uniformly distributed locations. Segment 2 follows inflexible choice behavior (I) and high values (H), while locations are also uniformly distributed. This demand setting represents an inner city, with one segment representing low-value, flexible demand (e.g. students, seniors) and the other segment representing high-value, inflexible demand (e.g. professionals).
<i>Center-clustered</i>	City center with clustered location distribution: Compared to <i>center-uniform</i> , the second demand segment is not distributed uniformly but lives in two clusters (a).
<i>Suburb-homogeneous</i>	Suburb with homogeneous basket values: Both demand segments provide low basket values but differ in their time choice flexibility. Also, they live in different areas in the region ((a) and (b)).

Fig. 6 Location distributions

6.2.1 Overall Results

Table 3 lists the resulting basket value per experiment, grouped by demand setting and dynamic slotting variant. Each column represents a problem scenario, defining the length of time slots (ts_2h is 0 or 1), time slot overlap ($ts_overlap$ is 0 or 1), region size (small region is 0 or 1), the number of vehicles (# of vehicles is 2 or 3), and the demand arrival rate λ . For instance, in the *center-uniform* demand setting, rout-IC achieves 1.292 times as much basket value as FCFS for a scenario with disjoint one-hour time slots, a small region, two vehicles, and arrival rate $\lambda = 0.3$. When decreasing the arrival rate to $\lambda = 0.27$, the gain of rout-IC over FCFS decreases to 25.9%. *pat-th⁺* can only surpass FCFS by 15% and 9% in these two scenarios. **FCFS is not competitive to any of the tested dynamic slotting approaches. Clearly, anticipating future value and/or route information is beneficial, improving revenue by up to 39%.**

The column *avg. yield* provides the overall earned basket value divided by the number of accepted orders, averaged over all problem scenarios for the demand segment setting. For instance, rout-IC's higher average ratio often performs better than *pat-th⁺* because it accepts more valuable orders (in addition to accepting more orders overall). The column *best perf.* indicates the number of scenarios where the approach earned the highest basket value. **The best approach to improve yield and its actual advantage strongly depend on the demand setting.** For instance, rout-IC's performs best in 17 out of 18 scenarios with the *center-uniform* demand setting, achieving in average 4.2% more basket value than the second best approach. In contrast, for the *center-clustered* setting, the holistic anticipative approach *pat-th⁺* achieves the best results in around 60% of cases. However, its advantage over the second best is lower. In the following, we further analyse results by considering different demand segment combinations and other problem setting characteristics.

6.2.2 Center-uniform Problem Settings

In this inner city setting, high-value, inflexible customers (e.g., professionals) and low-value, flexible customers (e.g., students) are distributed uniformly across

Table 3 (continued)

ts_{2h}	0	0	0	0	0	0	0	1 (0)	1 (0)	1 (0)	1 (0)	1 (0)	1 (0)	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)	1 (1)	14	1.066
$(ts_{overlap})$																					
pat-th+	1.254	1.184	1.298	1.242	1.393	1.327	1.18	1.104	1.222	1.146	1.346	1.291	1.267	1.211	1.296	1.235	1.393	1.372	1.264	14	1.066
Suburb-homogeneous																					
rout-IC	1.137	1.1	1.104	1.116	1.127	1.144	1.026	1.012	1	1.071	0.985	0.996	1.08	1.078	1.072	1.023	1.146	1.117	1.074	0	1.014
rpat-d0-#	1.094	1.061	1.075	1.063	1.124	1.106	1.054	1.037	1.034	1.032	1.078	1.058	1.098	1.079	1.067	1.043	1.105	1.097	1.073	0	1.005
rpat-d0-EC	1.146	1.098	1.135	1.118	1.19	1.154	1.09	1.052	1.07	1.051	1.124	1.1	1.135	1.103	1.102	1.075	1.17	1.147	1.114	0	1.005
rpat-d2-IC	1.131	1.1	1.106	1.107	1.168	1.159	1.07	1.034	1.007	1.066	1.013	1.054	1.096	1.077	1.089	1.035	1.173	1.138	1.09	0	1.008
rpat-d2-EC	1.164	1.132	1.137	1.124	1.212	1.179	1.101	1.071	1.073	1.057	1.127	1.102	1.15	1.12	1.11	1.084	1.17	1.156	1.126	1	1.005
rpat-d2-EC	1.141	1.109	1.11	1.095	1.16	1.133	1.095	1.071	1.078	1.062	1.117	1.105	1.145	1.117	1.096	1.071	1.14	1.141	1.11	0	1.012
pat	1.099	1.011	1.139	1.091	1.235	1.167	1	0.945	1.03	0.979	1.133	1.056	1.105	1.066	1.136	1.047	1.244	1.181	1.092	0	1.007
pat-th+	1.21	1.128	1.22	1.182	1.298	1.251	1.128	1.072	1.139	1.085	1.235	1.161	1.222	1.183	1.218	1.147	1.323	1.274	1.193	17	1.008

Table 4 *center-uniform*: Opportunity cost (2h-time slots, small region, 2 vehicles, $\lambda = 0.3$)

Opportunity cost		
$\delta_{st}\nabla_d = \beta_s + 0.4373\nabla_d + 0.1010i\nabla_d$	Time slot s	β_s
∇_d : Difference in remaining travel time (with and without new order for time slot s , normalized)	1	0.0889
	2	0.0252
	3	-0.0035
	4	0.0150
	5	0.0669
	6	0.0350

the region. **rout-IC achieves highest basket value results in 17 out of 18 scenarios.** *pat-th*⁺ is only competitive to rout-IC when three vehicles deliver order in the large region for an overlapping time slot set ($\lambda = 0.405$). In most scenarios, rout-IC accepts the highest number of orders and earns relatively high yield. Nevertheless, in several three-vehicle settings, *pat-th*⁺ accepted more orders and performs second best in terms of basket value.

To illustrate rout-IC's value anticipation, Table 4 shows the opportunity cost calculation for a 2-h-time slots scenario. The opportunity cost depend on the travel time utilized per order request (∇_d) and slightly reduce over time as the future expected demand decreases ($i\Delta_d$). Moreover, they differ across time slots (β_s), representing differences in demand volume and value.¹

To explain the behavior of *pat* variants, Fig. 7 visualizes different anticipatory schedule patterns for a popular 2h-time slot in a two-vehicle setting. The patterns are very heterogeneous, indicating a broad variety of delivery schedules over demand samples. Thus, the pool of relevant patterns is likely to shrink quickly during the order period. As a result, *pat* cannot dynamically adapt to accommodate requests with the highest basket values and has to assign capacity to less valuable requests within the reserved areas of the still relevant patterns. **While introducing theft-based**

Fig. 7 *center-uniform*: Different Anticipatory schedule patterns for a Popular Time Slot (2h-time slots, small region, 2 vehicles, $\lambda = 0.3$)

¹ Note that differences in coefficients can also result from normalization. For instance, Time Slot 1 has a higher maximum number of acceptable orders than later Time Slot 5, because the travel time from the depot is not considered as resource consumption. Thus, the higher time slot specific coefficient might not actually result in higher opportunity cost but compensate for the lower increase in $\ddot{x}_s$.

assignment helps, it cannot match **route-IC's flexibility**. Similarly, **rpat variants cannot profit from the favorability check**. From these, **rpat-d2-EC-TW** achieve the most basket value, as it provides the most flexibility in resource assignment when splitting anticipatory patterns by time slot.

Overall, route-IC succeeds via its anticipation of future value, but attempts to anticipate final routing schedules do not improve but rather impair results in the center-uniform setting.

6.2.3 Center-clustered Problem Settings

This inner city setting entails high-value, inflexible customers located in two clusters in opposite corners of the delivery region. Low-value, flexible customers are spread across the entire region. **In this setting, pat-th⁺ achieves the highest basket value for three vehicles, while route-IC performs best for two vehicles.** Compared to the *center-uniform* setting, the average yield does not vary much across approaches, indicating that the difference in performance is mostly caused by a difference in number of accepted orders.

Figure 8(a) illustrates how the accepted value accumulates over the order period for a two-vehicle scenario. Assigning routes to the two clusters is intuitive: high opportunity cost for popular time slots in route-IC ensure a favorable shape of delivery routes without the need to limit flexibility via anticipatory schedule patterns. FCFS accumulates value quickly and early but can only accept little more value in the last third of the order period. The other approaches accumulate value more slowly but continue to do so until the end of the order period. The development is very similar across all anticipative approaches. **For three vehicles, the guidance via patterns in pat-th⁺ significantly improves results, especially in the larger region.** Figure 8(b) exemplifies the value development for three vehicles. pat-th⁺ collects value fastest and until the end of the order period. rpat-d2-EC can

Fig. 8 *center-clustered*: Accumulated value development for a two vehicle (small region, $\lambda = 0.3$) and three vehicle scenario (large region, $\lambda = 0.45$) and 2h-time slots

accept less towards the end of the order period, mainly driven by the less efficient routes than the pat variants.

In two 2h-time slot scenarios, where rpat-d2-EC-TW achieves more basket value than rout-IC and pat-th⁺. rpat-d2-EC-TW represents a compromise in that it anticipates delivery schedules while allowing for the most flexible planning in terms of allowing neighboring areas and decomposing patterns by time slots. Moreover, rpat-d2-EC-TW performs better than rout-IC when considering three vehicles and two hour time slots, rendering the approach attractive when excluding pat-th⁺ to guarantee feasibility.

6.2.4 Suburb-homogenous Problem Settings

In this suburban setting, customers from different neighborhoods differ in the flexibility of their time slot choice. Basket value distributions are homogeneous over demand segments, such that there are no particularly valuable demand segments. pat-th⁺ performs best in most related scenarios, with rpat-d2-EC sometimes coming in second. When offering 1h-time slots and two vehicles cover the small region for a low arrival rate of 0.27, rpat-d2-EC even outperforms pat-th⁺. As segments do not differ in expected basket value, the relative impact of accepting more orders on basket value outcome is especially high compared to the other demand settings.

To analyze this, Fig. 9 illustrates the locations of accepted orders (black) and requests (white) for pat-th⁺ rpat-d2-EC on a demand sample where pat-th⁺ performs poorly. The figure provides the TOP routing schedule underlying pat-th⁺'s last relevant anticipatory schedule pattern. In this example, the pattern suggests to concentrate delivery in three clusters, two of which harbor flexible demand, and to accept only few orders in the fourth cluster. To achieve this, the routing schedule keeps one vehicle mostly in the flexible demand cluster next to the depot, while letting the other vehicle travel some detours to deliver orders from the inflexible customer segment before serving the second flexible demand cluster. While such a strategy is successful for an arrival rate of 0.3, the lower arrival rate of 0.27 decreases the

Fig. 9 Suburb-homogeneous: Accepted orders (black) from TOP schedule pattern (a) as compared to pat-th⁺ (b) and rpat-d2-EC (c) as resulting from the same set of actual order requests (1h-time slots, small region, two vehicles, $\lambda = 0.27$). Rejected orders are empty circles—the TOP schedule pattern does not store rejected orders

demand-capacity-ratio and makes it harder to assign the capacity of those clusters in the stochastic, online setting. In contrast, rpat-d2-EC considers a Boolean anticipatory pattern, which neglects the number of deliveries. Accordingly, it flexibly assigns more capacity to the second cluster of inflexible demand, here in the right lower corner. This example not only demonstrates the potential of rpat-d2-EC in low demand-capacity-ratio settings, but also motivates to consider aspects like expected demand-capacity-ratios when building anticipatory schedule patterns.

We conclude that this setting calls for anticipating final schedules to differentiate spatial clusters with flexible and inflexible time slots demand.

6.2.5 Variants of rpat: # Versus IC Versus EC

The rpat approach combines feasibility and favorability checks with several VFAM variants: Whereas the # variant VFAM only considers the number of accepted orders per time slot, the IC variant also features the additional time expected to deliver the order based on ad-hoc routes. The EC variant features the full expected travel time especially for settings with uniformly distributed locations or 1h-time slots, relying on myopic insertion cost in the IC variant outperforms sticking to previously defined plans. While estimates do not conform to final insertion cost, they directly relate to the dynamically built routes, which can significantly deviate from anticipatory schedules due to the inherent stochasticity. Nevertheless, **in many settings, anticipatory patterns and anticipated travel times successfully support building favorable routes.** In these cases, EC variants cannot outperform IC variants: the IC strategy of considering the remaining travel time instead of only considering the number of accepted orders (# variant) can be unnecessary or even counter-productive.

6.2.6 pat: Delivery Area Size and Effects From Theft-based Assignment

Next, we analyze the effect of area sizes on the number of infeasible orders and thefts in pat variants. Large areas enable more flexible capacity assignment, but increase the risk of infeasible orders. Given a specific area size, theft-based assignment introduces

Table 5 Infeasible orders and thefts for different area sizes (*center-uniform*: 2h-time slots, disjoint, small region, 2 vehicles, $\lambda = 0.27$)

Approach	Number of areas	Basket value (not adjusted, rel.)	Number accepted (rel.)	Number infeasible	Number theft 1	Number theft 2	Number theft 3
pat	64	0.576	0.565	0	–	–	–
pat	36	0.785	0.768	0	–	–	–
pat	16	0.928	0.902	0.01	–	–	–
pat-th ⁺	64	0.821	0.796	0	8.16	13.48	7.13
pat-th ⁺	36	1.012	0.983	0.03	15.1	8.56	10.16
pat-th ⁺	16	1.145	1.103	0.36	22.08	3.23	12.66

flexibility while keeping the resulting additional risk of in-feasibility low. To highlight this, we focus on the *center-uniform* demand setting.

Table 5 provides results from different ways of splitting the small region into areas. The number of infeasible orders and the number of thefts are given as absolute averages. Splitting the region into more, smaller areas reduces the number of accepted orders and basket value from the same dynamic slotting variant. However, implementing theft-based assignment in *pat-th*⁺ earns more on a grid of 36 areas than the variant without thefts earns on a 16 area grid. *pat-th*⁺ applies theft mechanisms more often for large areas, which thereby obtain even more flexibility. The number of thefts also increases with opportunity, as larger areas enable more potential theft patterns. For instance, theft mechanism 1 requires an accepted order to consider theft from a neighbor area. Larger areas both increase the probability of prior accepted orders in the current area and the probability that a neighbor has left-over capacity. **In conclusion, when decomposing the delivery region into areas, the delivery service provider can balance infeasibility and flexibility by carefully choosing area sizes for the specific problem setting.**

6.3 Large-scale Problem Scenarios

Having explored the effects from alternative dynamic slotting approaches and variants on limited problem scenarios, we show that the approaches scale efficiently for larger problem sizes based on the center-clustered setting. We concentrate on demand settings where route anticipation yielded better results, suggesting the additional runtime for solving the TOP is worthwhile. For settings with 6 vehicles, we use the large region size with 36 areas, an order horizon length of 1,500, and an arrival rate of 0.3 (450 expected requests) or 0.27 (405 expected requests). The settings with ten vehicles refer to a larger region of 25x25km (with 100 areas, causing approximately 7 minutes of diagonal travel time per area), an order horizon length of 2,500, and an arrival rate of 0.3 (750 expected requests) or 0.27 (675 expected requests).

Table 6 reports the results. **In all cases, pat variants with theft-based assignment outperform the other approaches by far.** Nevertheless, comparing rout and rpat results suggests careful benchmarking for center-clustered settings. Increasing

Table 6 Large-Scale Scenarios—Basket value scaled on FCFS, best performers marked **bold**

number of vehicles	6		10	
expected requests	405	450	675	750
rout - IC	1.187	1.212	1.192	1.18
rout - d0 - IC	1.175	1.204	1.218	1.25
rout - d2 - EC	1.181	1.201	1.218	1.261
pat	1.097	1.169	1.138	1.159
pat-th ⁺	1.271	1.327	1.358	1.387

Fig. 10 Sample locations for empirical scenario

the problem sizes increases the computational effort in the preparation phase. The highest average run-time of solving the TOP (10 vehicles, 750 requests) is about 22 minutes. As it can be run over night and in parallel, this should not be a problem for practical applications. In a single case with ten vehicles, training the VFAM took 105 minutes; all other settings required training run-times of less than one hour. **In sum, even the largest settings can be solved in less than three hours when parallelizing TOP solving.** The online run-time per order request linearly increases with the number of areas (pat) or requests (rout), still resulting in a maximum average run-time of below 1.9 ms (for pat) and 1.3 ms (for rout-IC).

6.4 Empirical Setting Scenarios

A final set of experiments features a setting derived from empirical data to examine the implications of applying dynamic slotting variants in a real-world setting. To that end, we obtained a log-normal basket value distribution and a customer location distribution from data of a German e-grocer. Moreover, we follow the maximum likelihood approach of [29] to estimate an MNL model from this data. As our industry partner does not charge delivery fees, the model estimation is restricted to alternative-specific, constant utilities. To anonymize the data, we

Table 7 Industry partner—
Estimated utility values

Alternative	Utility $\hat{u}_s$
10 am to 12 pm	0.346
12 pm to 2 pm	-0.984
2 pm to 4 pm	-1.623
4 pm to 6 pm	-1.272
6 pm to 8 pm	0
7 pm to 9 pm	-1.359
8 pm to 10 pm	-1.514

treat the duration of the order period and the arrival probability per time step as variables.

Table 7 shows the estimation results for one demand segment and demand for a Wednesday. The utility of the no-purchase option is set to 0. The base utility of 0.917 indicates the utility of the time slot from 6 to 8 pm, i.e. $\tilde{u}_{6-8} = 0.917$. The coefficients of the other time slots represent their differences in utility to this reference time slot, i.e. $\tilde{u}_s = 0.917 + \hat{u}_s$. They all significantly deviate from 0. The emerging customer behavior is similar to the flexible behavior featured in the synthetic scenarios, with a comparatively high utility for two time slots, 10 am to 12 pm and 6 pm to 8 pm. Fig. 10 provides sample locations for one run. For the experiments, we concentrate on an urban region of 11 km x 9 km, indicated by a box in the figure, and locate a depot in the north-east part of the region, as indicated by the orange triangle. In this region, the locations are not clearly clustered but also not uniformly distributed.

Table 8 provides the results of applying the dynamic slotting approaches to settings featuring three or four vehicles. **For demand-to-capacity-ratios exceeding 0.35, pat-th⁺ achieves highest basket value. Otherwise, rout outperforms the other approaches; anticipating final delivery schedules does not improve earned basket value.** These results confirm the observations from the synthetic scenarios for settings without clusters, as there are no spatial clusters that define more valuable or less flexible demand. As locations are not uniformly distributed, pat-th⁺ can apparently profit from the more efficient delivery schedules and thus, accept more customers for the higher demand-capacity-ratio settings than rout-IC.

6.5 Main Insights from the Computational Study

We conclude the computational study by extracting some insights to support decision makers in selecting an approach to implement for a specific application scenario:

- Anticipating final delivery schedules is effective when customer locations form spatial clusters that differ in value or choice behavior. When customers are dis-

Table 8 Empirical Setting—Basket value scaled on FCFs, best performer marked **bold**

number of vehicles	3	3	3	4	4	4
λ	0.315	0.35	0.385	0.45	0.5	0.55
rout-IC	1.11	1.152	1.21	1.139	1.186	1.187
rpat-d0-#	1.009	1.051	1.117	1.052	1.121	1.122
rpat-d0-EC	1.01	1.052	1.125	1.056	1.128	1.125
rpat-d2-IC	1.051	1.091	1.158	1.1	1.152	1.154
rpat-d2-EC	1.073	1.114	1.184	1.102	1.174	1.17
rpat-d2-EC-TW	1.097	1.147	1.213	1.122	1.191	1.188
pat	0.879	0.935	1.004	0.943	1.023	1.011
pat-th ⁺	1.066	1.135	1.224	1.115	1.217	1.203

tributed homogeneously, attempting to anticipate delivery schedules can even be counter-productive.

- Theft-based assignment improves pat's performance by letting it adjust to observed demand. The improvement is especially notable when customer locations are at least partly uniformly distributed.
- The pat variant with theft-based assignment performed best in the majority of scenarios, but risks accepting orders that cannot be feasibly delivered with the given set of vehicles. Alternatively, rpat variants with anticipated travel times allow immediate checks of feasibility and outperform rout-IC especially given clustered demand and 2h-time slots.
- All proposed solution approaches scale very well with problem size, such that even large problem scenarios can be prepared and dynamically solved. Computing anticipatory patterns and training the VFAM ahead of the order period only takes a few hours and enables computing the offer set within milliseconds on order arrival.
- While the study provides some general insights to guide method selection, it also reveals that there is no single best approach for all scenarios. Thus, it is crucial to extensively benchmark and carefully design algorithm components on settings derived from case-by-case empirical data to ensure a successful application in real-world applications.

7 Conclusion

This paper proposed to use a preparation phase ahead of the order period to anticipate delivery schedules and opportunity cost in dynamic slotting for attended home deliveries. We introduced three modular approaches that vary in their degree of reliance on anticipative information. The idea of preparing anticipative information before the start of the order period and of applying flexible mechanisms within the order period enables negligible run-time on request arrival. We presented a comprehensive computational study featuring both synthetic and empirically-validated problem scenarios. The study highlighted that the choice of the dynamic slotting approach for a specific application scenario should mainly depend on demand characteristics. While anticipating routes can be useful when customers live in spatial clusters with heterogeneous basket values or time slot choice behavior, it can be counter-productive in other cases. Theft-based variants can help to make customer acceptance more flexible, but also bear the risk of infeasible solutions. Therefore, planners must carefully benchmark and adjust dynamic slotting approaches to fit the specific scenario.

The approaches featured in this paper emphasise modularity, allowing for a straightforward exchange of individual solution components. As indicated in Table 1, the choice of implemented components can also be used to draw relations to solution approaches from slightly different problem settings. Nevertheless, the variants benchmarked in the computational study represent only a selection of conceivable options. As this paper focuses on analysing the value of anticipative

information in various scenarios, we kept the approaches' designs simple to ensure reasonable preparation times and comparable results.

As the model presented here assumes a fixed fleet of delivery vehicles and a fixed staff of drivers, it regards the variable cost per delivery as negligible. In that, it opposes a range of existing contributions as reviewed in Section 2. Applying a method that does not account for delivery-dependent costs to a situation where these are, in fact, relevant, can be expected to perform poorly, just as applying a method that does account for them when they are, in fact, irrelevant. Regarding the need to match model assumptions and real-life circumstances, it would be highly interesting to compare the performance of these dynamic pricing and dynamic slotting approaches in a variety of settings when variable cost does significantly affect earnings.

Further research could improve components, e.g. by developing approaches to solving the TOP formulation and solution to account for customer choice. Another opportunity lies in a more in-depth consideration on alternative approaches to measuring dissimilarity between anticipatory patterns. This might require a dedicated numerical study. Alternatives might also consider characterizing patterns as more or less robust or considering different granularities in the patterns. Further opportunities for further research include alternative customer choice models that may require a heuristic solution to the embedded assortment problem. Moreover, route anticipation could be improved by alternative distance measures or other strategies to use anticipatory patterns in an online policy. Improving on TOP and thus the resulting patterns is another possible entry point for further research. As each approach relies on the outcome of actual routing procedures, the approaches are also open for extension through new routing heuristics to accommodate problem variants featuring, e.g., time-dependent travel times or order-dependent vehicle types. We also recommend considering decomposing the region into spatial clusters before, similar to [28], depending on individual requirements on the length of the preparation phase. Alternatively, future research can concentrate on the handling of possibly infeasible orders in *pat*. For instance, an approach could explicitly anticipate risk of infeasibility and trade off additional basket value with cost of a cap or an additional vehicle as part of optimization. Moreover, with the best approximation approach depending on the scenario, we motivate research on a benchmarking tool which facilitates scientific rigor in AHD and increases its relevance for practice.

8 Appendix—Dissimilarity Measures

Let an order that falls into a yes-combination in the Boolean schedule pattern create a dissimilarity value of 0 to that pattern, regardless of the prior number of orders accepted in this combination. Otherwise, if the order's delivery area shares a border with a yes-combination, termed a neighbor area, the order's dissimilarity is set according to a parameter d^n . If the order is neither for a yes-combination nor for a neighbor, its dissimilarity is d^o . Figure 3 exemplifies

dissimilarity measure values for two Boolean schedule patterns. In the first example, three orders are accepted for yes-combinations of both patterns; the resulting dissimilarity measures are both 0. In the second example, an additional order is located in the upper right area. As this is a no-combination for pattern 2, its dissimilarity measure is d^n . For the fourth example, there is an order in a no-combination that is no neighbor for pattern 1, causing a dissimilarity measure of $d^n + d^o$. The lowest possible dissimilarity is 0, when all accepted orders fall within a yes-combination of the Boolean pattern.

9 Appendix—Notation

Compare Table 9.

Table 9 Notation

Notation	Explanation
$t = 1, \dots, T$	Discrete time step, order period length T
$\lambda \in [0, 1]$	Arrival probability per time step
$s \in \mathcal{S} = \{1, \dots, S\}$	Time slots with starting time b_s and ending time q_s
$j \in \mathcal{J} = \{1, \dots, J\}$	Order requests
$\mathcal{O} = \{1, \dots, O\}$	Accepted orders
r_j, r_l, z_r	Basket value
$l \in \mathcal{L} = \{1, \dots, L\}$	Customer segments, proportion $\eta_l \leq 1$
$a \in \mathcal{A} = \{1, \dots, A\}$	Delivery areas
$\mu_{la} \leq 1$	Probability that a request from customer segment l lives in area a
$P_{ls}(\mathcal{S}_{la})$	Probability that segment l chooses time slot s for offer set $\mathcal{S}_{la}$
$\tilde{u}_{ls}$	Segment-specific utility constant for time slot s
$u_{js} = \tilde{u}_{ljs} + \epsilon_{js}$	Request j 's utility for time slot s
$X = [x_{sa}, \dots, x_{SA}]$	Vector of number of accepted orders x_{sa} per time slot s and area a
$\mathcal{X}$	Set of states that can be feasibly fulfilled by the given vehicles
$m \in \mathcal{M} = \{1, \dots, M\}$	Vehicles
$\mathcal{F}_a(X) \subseteq \mathcal{S}$	Set of feasible time slots
$V_t(X)$	Maximal expected value for state X from time t until the end of the order period.
δ_{xtas}	Opportunity cost for state X in t and a potential order for time slot s in area a
ζ_{sj}	GRILS: Minimum required time shift towards the end of routes to insert j for time slot s
$\mathcal{C}$	Candidate set of feasible (rpat:and favorable) time slots
$\hat{V}_t(\sigma(\mathcal{O}_t))$	Approximated value function value for features $\sigma(\mathcal{O}_t)$ of accepted orders
$V_t(\mathcal{O}_t)$	Observed value function value
$\hat{\delta}_{t\sigma(\mathcal{O}_t)s}$	Approximated opportunity cost for time step t , features $\sigma(\mathcal{O}_t)$ of accepted orders $\mathcal{O}_t$ and time slot s
$L(\mathcal{O}_t)$	Loss function
$R_t(\mathcal{O}_t)$	Observed reward in time step t for accepted orders $\mathcal{O}_t$

Table 9 (continued)

Notation	Explanation
α	Learning rate
Q	Annealing parameter Q
ω	Momentum weight
t	Remaining time in the order period (normalized)
$\tilde{d}, \nabla_{\tilde{d}s}$	Remaining travel time (normalized), change in remaining travel time
$\tilde{x}_s$	Number of accepted orders for time slot s (normalized)
$\beta_0, \beta_s, \beta_d, \beta_t, \beta_{xt}$	Coefficients of a linear value function approximation model
d^n, d^o	Distance of an order to a Boolean pattern, either the order's no-combination is a neighbor of yes-combination (d^n) or no neighbor (d^o)
ϕ, ϕ^{max}	Difference in aggregated distance measure, maximum acceptable difference in aggregated distance measure

10 Appendix—Team Orienteering Problem with Multiple Time Windows

When modelling deterministic dynamic slotting as a TOP-MTW, vertices correspond to order requests $\mathcal{J}$. Moreover, the depot constitutes both vertex 0 and $J+1$, as it is the obligatory start and end of each of the M tours. Every vertex j can be visited during a subset of time slots $\mathcal{S}_{u_{js} \geq u_{j0}} \subseteq \{1, \dots, S\}$, which are given by the consideration set of slots the customer prefers to the no-purchase option, i.e. $u_{js} \geq u_{j0}$ for all $s \in \mathcal{S}_{u_{js} \geq u_{j0}}$. We adjust the mathematical model of the problem given by [23] to our notation and setting, especially in order to capture the inherent stochasticity of demand. There results the following integer program.

Let f_{ij} be the shortest path between customers i and j , in terms of travel time. The starting time of serving customer i in tour m is e_{im} , while h_i is the service time of customer i . W is a large constant. The decision variables include binary variable x_{ijm} , which is 1 if visiting customer j after customer i in tour m , as well as binary variable y_{jms} , which is 1 if visiting customer j in time window s and tour m . The following objective function maximizes the total score collected.

$$\max \sum_{m=1}^M \sum_{s=1}^S \sum_{j \in \mathcal{J}_s} r_j y_{jms} \quad (9)$$

$$\sum_{m=1}^M \sum_{j=1}^{J+1} x_{1jm} = \sum_{m=1}^M \sum_{i=0}^J x_{i(J+1)m} = M \quad (10)$$

$$\sum_{i=0}^J x_{ikm} = \sum_{j=1}^{J+1} x_{kjm} = \sum_{s \in C_j} y_{ksm}, \forall k \in \mathcal{J}, \forall m \in \mathcal{M} \quad (11)$$

$$\sum_{m=1}^M \sum_{s \in S_j} y_{jsm} \leq 1, \forall j \in \mathcal{J} \quad (12)$$

$$e_{im} + h_i + f_{ij} - e_{jm} \leq W(1 - x_{ijm}), \forall i, j \in \{0, \dots, J+1\}; \forall m \in \mathcal{M} \quad (13)$$

$$\exists s \in \mathcal{S}_{u_{js} \geq u_{j0}} : o_s \leq g_{jm} \leq q_s, \forall j \in \mathcal{J}; \forall m \in \mathcal{M} \quad (14)$$

$$x_{ijm}, y_{ism} \in \{0, 1\}; \forall i, j \in \{0, \dots, J+1\}; \forall m \in \mathcal{M}; \forall s \in \mathcal{S} \quad (15)$$

$$v_{as} \geq 0, \forall a \in \mathcal{A}; \forall s \in \mathcal{S} \quad (16)$$

In this linear program, constraints 10 ensure that all tours start and end in the depot. Constraints 11 guarantee the connectivity of each tour, while constraints 12 ensure that every customer is only visited at most once. Constraints 13 ensure the time-related feasibility of the tours, i.e. the service time of an order j cannot start earlier than its predecessor was served and the vehicle travelled between the two locations. Moreover, constraints 14 determine that the customers are served within one of their considered time windows (model adapted from [23]).

11 Appendix—Dynamic Slotting Approaches

11.1 rout and rpat – Technical Details

Stochastic gradient descent update The stochastic gradient descent update for the linear model looks as follows.

Let ω be the momentum weight and $\psi_g^{(n-1)}$ be the last update of feature g . $L^n = \frac{1}{2}(V(\mathcal{O}) - \hat{V}^n(\sigma(\mathcal{O})))^2$ is the squared loss function. Then a stochastic gradient update of current feature coefficient β_g^n looks as Equation (17).

$$\beta_g^n = \beta_g^{(n-1)} - \alpha^n((1 - \omega)\frac{\partial L^n}{\partial \beta_g} + \omega\psi_g^{(n-1)}) \quad (17)$$

Afterwards, the last update value is updated as $\psi_g^n = (1 - \omega)\frac{\partial L^n(Y)}{\partial \beta_g} + \omega\psi_g^{(n-1)}$.

Initialization We initialize the model by treating TOP results as sample trajectories of an order horizon.² As TOP routing schedules are results of a deterministic, full information problem setting, such an approach likely leads to overestimated opportunity cost. Therefore, we use a discounting factor γ and weigh the orders' values with the maximum probability $\hat{P}_{ls} = P_{ls}(\{0, s\})$ of choosing the time window s . These measures allow more realistic initial approximation values and facilitate learning. As opposed to this initialization, the online policy does not treat future

² rout can completely omit the TOP results by using historical routing schedules. In this study, TOP results ensure comparable outcomes

rewards explicitly as less valuable. The uncertainty is implicitly represented in the opportunity cost, as they represent possible future outcomes. The resulting observations serve as input for a multiple regression approach to obtain initial coefficients.

11.2 pat's Optimization Problem—Maximize Value per Anticipatory Pattern

Opportunity cost approximations in pat's online policy are based on aggregated value function approximations over patterns. In this section, we define the underlying maximization problem of valuably assigning capacity per pattern, that is the value function which pat has to approximate.

We decompose the optimization problem on the level of delivery areas. We define the optimization model for this decomposition, omitting the area index in the following. A complete value function approximation is the sum over all area level approximations.

We model the problem of optimizing the offer set per request to maximise the overall value of accepted orders as follows: State vector variable $X = [x_1, x_2, \dots, x_S]$ represents the number of already accepted orders per time slot. The capacity per time slot, obtained from the anticipatory pattern, is $C = [c_1, c_2, \dots, c_S]$. The offer set is a subset of the set $\mathcal{S}_x$ of time windows with left-over capacity $c_s - x_s > 0$, i.e. $\mathcal{S}_j \subseteq \mathcal{S}_x \subseteq \mathcal{S}$.

The following value function describes the maximum expected, remaining value for an order request j living in area a and arriving at time $1 \leq t \leq T$:

$$V_{ij}(X) = \max\{P_{ij0}(S_j)V_{t+1}(X) + \sum_{s \in S_j} P_{ijs}(S_j)(r_j + V_{t+1}(X + 1_{as}))\} \quad (18)$$

$\sum_{n \in S_j} P_{ijs}(S_j)(r_j + V_{t+1}(X + I^s))$ denotes the expected value if customer j accepts any offered time slot. $P_{ij0}(S_j)V_{t+1}(X)$ represents the expected value if customer j aborts the order and the number of accepted orders X is not increased.

In this, $V_t(X)$ defines the maximal expected value for accepted orders X at time $T \geq t \geq 1$:

$$V_t(X) = (1 - \lambda)V_{t+1}(X) + \lambda \sum_{l=1}^L \eta_l V_{tl}(X) \quad (19)$$

with the boundary conditions

$$V_{wt}(C) = 0, \text{ for } 1 \leq t \leq T \text{ and } V_0(X) = 0, \text{ for any number of accepted orders } X. \quad (20)$$

Thus, when as many orders were accepted per time slot as capacity available, the expected maximal value is 0. Likewise, when the order horizon has ended at $t = 0$, the expected maximal value is also 0.

In Equation (19), $1 - \lambda$ denotes the probability of no request arriving in t , so that the capacity is not reduced for $t + 1$. If a request from customer segment l

arrives, the segment-specific $V_{il}(X)$ results. $V_{il}(X)$ defines the maximum expected value for the number of accepted orders X for an arrival from segment l for $1 \leq t \leq T$:

$$V_{il}(X) = \max_{S_{it} \subseteq S_x} \{P_{l0}(S_{it})V_{(t+1)}(X) + \sum_{s \in S_{it}} P_{ls}(S_{it})(r_l + V_{(t+1)}(X + 1_{as}))\} \quad (21)$$

The structure of $V_{il}(X)$ is analog to request-specific $V_{ij}(X)$. However, the value r_l depends on the expected values for customer segment l . Consequentially, the offer set for an arriving request can deviate from the offer set reserved for expected requests from the same segment.

11.3 pat: Technical Details

Activation function in ANN. pat uses a tanh activation function. Thus, with $z_j = \sum_{i \in I} w_{ij}s_i + b_j$ for node j and input signals s_i from all nodes $i \in I$ of the previous layer and respective weights w_{ij} and bias b_j , we obtain activation function $h(z_j)$.

$$h(z_j) = \frac{e^{z_j} - e^{-z_j}}{e^{z_j} + e^{-z_j}}$$

Stochastic gradient descent update. We update the ANN weights over time steps $0 \leq t \leq T$ and per delivery area via stochastic gradient descent steps and backpropagation. Learning rate α and momentum weight ω control learning. Thus, weight w_{ij} between node i and j is updated as follows.

$$w_{ij}^n = w_{ij}^{n-1} - \alpha((1.0 - \omega)s_i q_j + \omega \psi_{ij}^{n-1})$$

with

$$\begin{aligned} q_J &= (y - h(z_J))(1 - h(z_J)^2) \text{ for output node } J \text{ and} \\ q_j &= w_{jJ} q_J (1 - h(z_j)^2) \text{ for node } j \text{ of the hidden layer.} \end{aligned} \quad (22)$$

In this, q_j represents the error propagated back for node j and ψ_{ij}^{n-1} is the previous update value. Thus, $\psi_{ij}^n = (1.0 - \omega)s_i e_j + \omega \psi_{ij}^{n-1}$.

Initialization The initialization with observations from TOP results employs the same kind of Monte Carlo updates as the learning phase Table 10.

12 Appendix—Computational Study

12.1 Time Slot Choice Models

Funding Information Open access funding provided by University of Vienna. This research was supported by a grant from the German Research Foundation (DFG, Grant No. CL605/2-1 and EH449/1-1).

Data Availability We provide access to the code and scenarios underlying the approaches and study at <https://github.com/SimlabCreator/silful>.

Table 10 Time slots and utilities

Number	1-hour time slots			2-hour time slots			2-hour time slots (overlapping)		
	Time	Utility-Flexible (F)	Utility-Inflexible (I)	Time	Number	Utility-Flexible (F)	Utility-Inflexible (I)	Time	Number
1	10am-11am	ln(1)	ln(4)	10am-12pm	1	ln(1)	ln(4)	10am-12pm	1
2	11am-12pm	ln(1)	ln(4)	12pm-2pm	2	ln(1)		11am-1pm	2
3	12pm-1pm	ln(1)		2pm-4pm	3	ln(0.5)		12pm-2pm	3
4	1pm-2pm	ln(1)		4pm-6pm	4	ln(1)		1pm-3pm	4
5	2pm-3pm	ln(0.5)		6pm-8pm	5	ln(1.5)	ln(4)	2pm-4pm	5
6	3pm-4pm	ln(0.5)		8pm-10pm	6	ln(1)		3pm-5pm	6
7	4pm-5pm	ln(1)						4pm-6pm	7
8	5pm-6pm	ln(1)	ln(0.5)					5pm-7pm	8
9	6pm-7pm	ln(1.5)	ln(4)					6pm-8pm	9
10	7pm-8pm	ln(1.5)	ln(4)					7pm-9pm	10
11	8pm-9pm	ln(1)						8pm-10pm	11
12	9pm-10pm	ln(1)							

Declarations

Conflict of Interest The authors declare that they have no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Agatz N, Campbell AM, Fleischmann M, Savelsbergh M (2011) Time slot management in attended home delivery. *Transp Sci* 45(3):435–449
2. Agatz N, Campbell AM, Fleischmann M, Van Nunen J, Savelsbergh M (2013) Revenue management opportunities for internet retailers. *J Revenue Pricing Manag* 12(2):128–138
3. Asdemir K, Jacob VS, Krishnan R (2009) Dynamic pricing of multiple home delivery options. *Eur J Oper Res* 196(1):246–257
4. Campbell AM, Savelsbergh M (2006) Incentive schemes for attended home delivery services. *Transp Sci* 40(3):327–341
5. Campbell AM, Savelsbergh MW (2005) Decision support for consumer direct grocery initiatives. *Transp Sci* 39(3):313–327
6. Chatwin RE (1998) Multiperiod airline overbooking with a single fare class. *Oper Res* 46(6):805–819
7. Cleophas C, Ehmke JF (2014) When are deliveries profitable? *Business & Information Systems Engineering* 6(3):153–163
8. Davis JM, Gallego G, Topaloglu H (2014) Assortment optimization under variants of the nested logit model. *Oper Res* 62(2):250–273
9. Ehmke JF, Campbell AM (2014) Customer acceptance mechanisms for home deliveries in metropolitan areas. *Eur J Oper Res* 233(1):193–207
10. Hübner A, Kuhn H, Wollenburg J (2016) Last mile fulfilment and distribution in omni-channel grocery retailing: a strategic planning framework. *Int J Retail Distrib Manag* 44(3):228–247
11. Klein R, Mackert J, Neugebauer M, Steinhardt CA (2018) model-based approximation of opportunity cost for dynamic pricing in attended home delivery. *OR Spectrum* 40:969–996
12. Klein R, Neugebauer M, Ratkovitch D, Steinhardt C (2017) Differentiated time slot pricing under routing considerations in attended home delivery. *Transp Sci* 53(1):236–255
13. Koch S, Klein R (2017) Time window budgets for anticipation in attended home delivery. *SSRN Electronic Journal*
14. Köhler C, Ehmke JF, Campbell AM (2019) Flexible time window management for attended home deliveries. *Omega* in Press
15. Kunnumkal S (2014) Randomization approaches for network revenue management with customer choice behavior. *Production and Operations Management* 23(9):1617–1633
16. Lang M, Cleophas C (2020) Establishing an extendable benchmarking framework for e-fulfilment. In *Proceedings of the 53rd Hawaii International Conference on System Sciences*
17. Lin II, Mahmassani HS (2002) Can online grocers deliver?: Some logistics considerations. *Transp Res Rec* 1817(1):17–24
18. Mackert J (2019) Choice-based dynamic time slot management in attended home delivery. *Comp Ind Eng* 129:333–345
19. Meissner J, Strauss A, Talluri K (2013) An enhanced concave program relaxation for choice network revenue management. *Prod Oper Manag* 22, 1 (2013), 71–87

20. Melacini M, Perotti S, Rasini M, Tappia E (2018) E-fulfilment and distribution in omni-channel retailing: a systematic literature review. *Int J Phys Distrib Logist Manag* 48(4):391–414
21. Powell WB (2007) *Approximate Dynamic Programming: Solving the curses of dimensionality*, vol. 703. John Wiley & Sons Inc, Hoboken, New Jersey
22. Savelsbergh MW (1985) Local search in routing problems with time windows. *Ann Oper Res* 4(1):285–305
23. Souffriau W, Vansteenwegen P, Vanden Berghe G, Van Oudheusden D (2013) The multiconstraint team orienteering problem with multiple time windows. *Transp Sci* 47(1):53–63
24. Sutton RS, Barto AG (2018) *Reinforcement learning: An introduction*. MIT press, Cambridge, MA
25. Ulmer MW, Mattfeld DC, Köster F (2017) Budgeting time for dynamic vehicle routing with stochastic customer requests. *Transp Sci* 52(1):20–37
26. Ulmer MW, Thomas BW (2019) Meso-parametric value function approximation for dynamic customer acceptances in delivery routing. *European Journal of Operational Research* in Press
27. Vansteenwegen P, Souffriau W, Berghe GV, Van Oudheusden D (2009) Iterated local search for the team orienteering problem with time windows. *Comp Oper Res* 36(12):3281–3290
28. Yang X, Strauss AK (2017) An approximate dynamic programming approach to attended home delivery management. *Eur J Oper Res* 263(3):935–945
29. Yang X, Strauss AK, Currie CS, Eglese R (2016) Choice-based demand management and vehicle routing in e-fulfillment. *Transp Sci* 50(2):473–488
30. Zhang D, Cooper WL (2005) Revenue management for parallel flights with customer-choice behavior. *Oper Res* 53(3):415–431