Supervised Learning for Arrival Time Estimations in Restaurant Meal Delivery

Preprint · August 2020

CITATIONS
0

READS
438

2 authors:

Florentin Hildebrandt
Technische Universität Braunschweig
1 PUBLICATION **0** CITATIONS
SEE PROFILE

Marlin Ulmer
Technische Universität Braunschweig
44 PUBLICATIONS **338** CITATIONS
SEE PROFILE

Supervised Learning for Arrival Time Estimations in Restaurant Meal Delivery

Florentin D. Hildebrandt Marlin W. Ulmer

August 2020

Abstract

Restaurant meal delivery companies have begun to provide customers with meal arrival time estimations to inform the customers' selection. Accurate estimations increase customer experience while inaccurate estimations may lead to dissatisfaction. Estimating arrival times is challenging because of uncertainty in both delivery and meal preparation process. To account for both processes, we present an offline as well as an online-offline estimation approach. Our offline method uses supervised learning to map state features directly to expected arrival times. Our online-offline method pairs online simulations with an offline approximation of the underlying assignment and routing policy; again achieved via supervised learning. Our computational study shows that both methods perform comparably to a full near-optimal online simulation at a fraction of the computational time. We also present an extensive analysis on how arrival time estimation changes the experience for customers, restaurants, and the platform.

Keywords: Stochastic Dynamic Vehicle Routing, Machine Learning, Arrival Time Predictions, Restaurant Meal Delivery

1 Introduction

Online food delivery is a billion-dollar industry. In 2019, the global annual revenue was US$ 107,445m with an average annual growth-rate of 12.2%. The majority of revenue falls into the segment of platform-to-consumer delivery (Statista Survey 2020). Platform-to-consumer

delivery services like UberEats, GrubHub and DoorDash mediate between customers and restaurants. They allow customers to order from participating restaurants and subsequently carry out the delivery themselves. Outsourcing logistics from restaurants to platform-to-consumer delivery services expands restaurants' market without requiring an own fleet. In a competitive market, each platform-to-consumer delivery service tries to get an edge over their competitors by binding new customers to their platform. Improving customer satisfaction is crucial to achieve this goal. The delivery process has a strong influence on customer satisfaction. Complaints about late deliveries are abundant, but there are also ample reviews that critique deliveries earlier than communicated. This is coherent with the literature where customer satisfaction depends on both, service quality and service expectation (Reimann et al. 2008). In the context of restaurant meal delivery, this means that not only fast delivery but also accurate meal arrival time estimation are important. In fact, the deviance of expected time of service is often a better predictor of customer satisfaction than actual time of service is (Thompson et al. 1996; Davis and Heineke 1998).

Estimating arrival times is challenging due to uncertainty in preparation and delivery process. The preparation time for an order and the time for parking and delivery may vary. Further, future orders may be bundled with existing delivery tours, for example, when an order is placed at the same restaurant only a short time later. A delay in preparation or delivery process may not only delay the respective order but may lead to propagation to other orders assigned to the driver. The bundling with a future order may further shift the arrival time.

Suitable methods should anticipate the mentioned uncertainty in a fast manner to provide customers with arrival time estimations in real time. Foreseeing delay requires to make the most of available information raising the need for detailed arrival time estimation methods. Unfortunately, accuracy often comes at the cost of runtime and the margin for a favorable runtime-accuracy trade-off is thin: Customers typically expect arrival times for multiple restaurants the very second they visit the website of a restaurant-meal-delivery platform. To satisfy this constraint, prediction time per restaurant must not exceed a fraction of a second.

To overcome these challenges, we propose two methods: An offline method that is trained prior to being employed and an online-offline that combines real-time simulation with offline

approximations. The methods address aforementioned uncertainties to derive accurate arrival time estimations while satisfying the given runtime constraints. Our offline method is a supervised learning model. It predicts arrival times based on state features by means of gradient boosted decision trees (GBDTs). The method relies only on historical data and does not require a detailed simulator. In contrast, our online-offline method is inspired by full online simulations of delivery process, meal preparation process, future demand, and routing. Full online simulations are highly accurate but infeasible due to the high computational costs of simulating routing decisions. Our online-offline method exploits an offline supervised learning of the runtime-expensive routing and assignment policy to perform detailed online simulations in real time. The offline routing approximation is achieved by learning the routing and assignment policy supervised in a simulator with the help of a deep neural network (DNN).

We compare our offline and our online-offline methods against two benchmark methods. The first benchmark is myopic, but fast, and estimates arrival times by assuming static and deterministic information. The second benchmark method performs full online simulations with high runtimes. We show that our offline method and our online-offline method always outperform the myopic benchmark method by a wide margin. The proposed methods come close to the lower bound of full online simulations without noticeable runtimes.

We use our offline method for a detailed managerial analysis of the effect of communicating arrival times to customers. We derive the following insights:

1. Our trained solution method remains accurate when business grows and data changes.

2. Customer flexibility in wait time and restaurant selection has a major impact on the overall number of orders per day.

3. Customers in the outskirts of the service area face long wait times and often reject to order. Proximity of a restaurant to densely populated areas is a strong predictor of the amount of orders the restaurant receives.

4. Advertising restaurants does not lead to workload congestion and is most effective for restaurants with a below average amount of orders.

5. A decrease in reliability of preparation times for a restaurant is linked to a decrease in orders and arrival time estimation quality for the restaurant as well as the overall performance of the restaurant-meal-delivery platform.

The contributions of the paper address model and methodology. To the best of our knowledge, our paper is among the first to estimate arrival times for a dynamic pickup and delivery problem. We present a new and unified mathematical model, capturing the interdependencies of arrival time estimation and the restaurant meal delivery problem. Our offline and online-offline methods perform near-optimally despite their low runtime. Our offline method is robust to changes in fleet, restaurants, and demand. Our online-offline method is not limited to arrival time estimations and is therefore a general contribution to the field of dynamic vehicle routing. The proposed approach to learn a routing policy supervised enables swift online simulation of dynamic routing problems that were previously limited by the computational burden of routing decisions. To the best of our knowledge, this idea is novel in the vehicle routing literature.

The paper is outlined as follows: In Section 2, we present and discuss related literature. In Section 3, we formally define the estimating arrival times for restaurant meal delivery problem (EAT). In Section 4, we motivate and present our offline and online-offline method. In Section 5, we provide details on our computational study and discuss the solution quality of our proposed methods. In Section 6, we analyze the impact of communicating arrival times to customers from the point of view of meal delivery platform, customer, and restaurant. In Section 7, we conclude the paper with a summary of our results and an outlook on future work.

2 Literature Review

In this section, we present related literature. Our problem combines three components: restaurant meal delivery, arrival time estimation, and supervised machine learning. There is only limited work in this domain. We first present the three most relevant works in Subsection 2.1 and then give an overview of works on the individual components in Subsections 2.2-2.4.

2.1 Most Related Work

The work most closely related to ours is that of Liu et al. (2018), Ulmer and Thomas (2019), and Zhu et al. (2020). Liu et al. (2018) consider meal delivery from a single restaurant from where drivers pickup meals for a set of customers. They focus on the challenge that the navigation and customer sequencing of drivers are unknown. Thus, the costs of assignment decisions are uncertain as well. Liu et al. (2018) predict the drivers' routes in a data-driven approach and solve the order assignment problem. Our work assumes the driver routing to be known up to uncertainty in service and preparation time as well as bundling operations with future requests. The work by Liu et al. (2018) on travel time estimation complements our work and can be integrated in our framework. Ulmer and Thomas (2019) consider the home service problem where a service provider assigns time windows to customers and subsequently serves them with a single vehicle. The problem comes with uncertainty in travel and service times as well as uncertainty in bundling due to dynamic requests. Ulmer and Thomas (2019) approximate arrival times by first aggregating the state space and second assigning mean arrival times derived from offline simulations to the resulting partitions. The work by Ulmer and Thomas (2019) considers a single vehicle, no pickups or dynamic routing, and no synchronisation of parallel processes in routing and meal preparation. The method of Ulmer and Thomas (2019) also employs temporal features only. We show in our analysis that spatial features are crucial for accurate arrival time predictions for the RMDP. Zhu et al. (2020) predict meal arrival times using a deep neural network trained on historical data of a Chinese restaurant-meal-delivery platform. Zhu et al. (2020) are concerned with the broader picture, integrating several aspects such as meteorological data, individual courier performance, and time to prepare specific dishes. While their historical arrival times are influenced by uncertainty in travel time, bundling, and synchronisation, they do not model them explicitly. They also neglect any routing information and forecasts of future demands. As we will later show, both are very valuable for accurate arrival time estimations.

2.2 Restaurant Meal Delivery

The restaurant meal delivery problem (RMDP) was recently introduced by Ulmer et al. (2020) and by Reyes et al. (2018). In the RMDP, a fleet of delivery vehicles serves dynamic customer requests that occur over the course of the day. Customers choose from a list of restaurants and place their order. The restaurant meal delivery service assigns each order to a driver, who picks up the meal at the restaurant and delivers it to the customer. The RMDP belongs to the class of dynamic pickup and delivery problems (dPDP), see, for example, Berbeglia et al. (2010). The additional challenge of the RMDP is characterized by the unique stochasticity in delivery- and meal preparation process as well as the urgency of dynamic requests. The limited work on RMDP analyses how to minimize delivery times (Reyes et al. 2018; Yildiz and Savelsbergh 2019), avoid delay (Ulmer et al. 2020), or how to schedule drivers in case the workforce is uncertain (Dai and Liu 2019; Ulmer and Savelsbergh 2020). None of the works are concerned with estimations of arrival times and their consequences.

2.3 Arrival Time Estimation

Arrival time estimation for transportation and logistics has been studied in the literature. There is work on traffic information systems that estimate travel times for given road segments; public transit systems in which arrival times are predicted for fixed routes; lead time prediction for industrial processes; and stochastic vehicle routing with time windows (SVRPTW) where time estimations are required to construct time windows and routes in accordance to each other.

Tab. 1 presents an overview of work that address arrival time estimations in logistics and transportation. The first part of the table presents the most related work that we discussed in Section 2. The second part gives a general overview on arrival time estimations. We classify the literature with regards to their considered uncertainty and their solution approach. The first column states the author, year, and field of application of the paper. The column "Routing" indicates whether time estimations are based on vehicle routes. For instance, lead time prediction (Polato et al. 2014; Lingitz et al. 2018) focuses on a manufacturing environment that does not necessarily include routes. We put the checkmark in parenthe-

Table 1 Classification of Literature on Arrival Time Estimations

Literature & Application	Routing	Uncertainty			Solution Approach	
		Travel/Service Times	Bundling	Processing Times	Offline	Online
Liu et al. (2018) (Restaurant Meal Delivery)	✓	✓	(✓)		✓	
Ulmer and Thomas (2019) (Home Service)	✓	✓	✓		✓	
Zhu et al. (2020) (Meal Arrival Time Prediction)	(✓)	(✓)	(✓)	(✓)	✓	
Chen et al. (2004) (Public Transit System)	✓	✓			✓	✓
Chu et al. (2005) (Travel Time Prediction)	(✓)	✓				✓
Van Lint et al. (2005) (Traffic Information System)	(✓)	✓			✓	
Lee (2009) (Travel Time Prediction)	(✓)	✓			✓	
Lee et al. (2009) (Travel Time Prediction)	✓	✓			✓	✓
Li et al. (2010) (SVRPTW)	✓	✓				✓
Zhang et al. (2013) (SVRPTW)	✓	✓			✓	
Polato et al. (2014) (Lead Time Prediction)				✓	✓	✓
Kim (2017) (Travel Time Prediction)	✓	✓			✓	✓
Lingitz et al. (2018) (Lead Time Prediction)				✓	✓	
Poschmann et al. (2019) (Maritime Transport Chain)	(✓)	✓		✓	✓	
Wu and Wu (2019) (Package Delivery System)	✓	✓			✓	
Vareias et al. (2019) (SVRPTW)	✓	✓			✓	
Prokhorchuk et al. (2019) (Traffic Information System)	(✓)	✓				✓
Bertsimas et al. (2019) (Traffic Information System)	(✓)	✓			✓	
Huang et al. (2020) (Travel Time Prediction)	(✓)	✓			✓	
Florio et al. (2020) (Travel Time Prediction)	(✓)	✓			✓	✓
Hoogeboom et al. (2020) (SVRPTW)	✓	✓				✓
Our work (Restaurant Meal Delivery)	✓	✓	✓	✓	✓	✓

ses if routes consist of a trip between a single origin and destination; i.e. when predicting freeway travel times for single road segments (Lee 2009). The column "Uncertainty" indi-

cates whether uncertainty in travel/service times, uncertainty in bundling with future orders or jobs, or uncertainty in processing times are considered. Uncertainty in travel/service times is prominent in urban areas where traffic congestion is frequent and parking spots are scarce or when service times are volatile; uncertainty in bundling occurs when customers request dynamically and vehicles are allowed to bundle requests in the route such that existing requests are shifted to a later time; uncertainty in processing times is caused by the synchronisation of two stochastic processes, i.e. the synchronisation of delivery process and meal preparation process. Uncertainty in processing times also occurs in intermodal logistic networks, where processes of different stakeholders have to be synchronized along the transport chain (Poschmann et al. 2019). The column "Solution Approach" indicates whether the authors employed an offline or an online method. Offline methods conduct the majority of calculations prior to their deployment. In contrast, online methods conduct calculations on-the-fly.

Following our classification in Tab. 1, we analyze the literature with regards to the underlying model and the solution method. From the point of view of the underlying model, we are interested in time estimation for routing problems. We first consider the arrival time estimation for SVRPTW. As the RMDP, the SVPRTW combines temporal estimations with routing but lacks the dynamicity of the RMDP. The majority of work on SVRPTW focuses on stochastic service and travel times but does not consider dynamic requests. Li et al. (2010) consider a stochastic vehicle routing problem with given time windows. When designing routes, time windows have to be satisfied with a fixed probability. Li et al. (2010) check whether a route is feasible by directly approximating the unknown expected arrival times by means of online simulations. In their work, they use exact information on the distribution of travel and service times. However, Li et al. (2010) indicate that online simulation is computationally too demanding for practical application. Zhang et al. (2013) minimize the carrier's total costs while guaranteeing a minimum on-time arrival probability. They estimate arrival times by a sum of expected service and travel times to calculate the on-time arrival probability. Vareias et al. (2019) design minimum-cost routes and then assign time windows that minimize the associated expected penalty costs. Vareias et al. (2019) only consider stochastic travel times and shape arrival times as the Cartesian product of disruption

scenarios for arcs traveled to reach the customer. Hoogeboom et al. (2020) simultaneously construct routes and time windows that minimize the expected travel time and the risk that time windows are violated. They estimate arrival times based on mean travel times for each arc. The solution methods of Zhang et al. (2013), Vareias et al. (2019), and Hoogeboom et al. (2020) are not designed to handle uncertainty in bundling or synchronisation as present in our problem setting.

None of the works reflect the unique time estimation challenge posed by restaurant meal delivery. In the RMDP, we face stochastic parking and delivery times, we face uncertainty in bundling orders from the same or adjacent restaurants, and we face wait times due to the synchronisation of delivery process with meal preparation process.

Next, we analyze the literature with regards to the solution method. We propose an offline and an online-offline method to estimate arrival times for the RMDP. In our offline approach, we map aggregated states to expected arrival times by means of gradient boosted decision trees. Offline approaches are frequently used in time estimation for transportation and logistics. We first discuss offline methods to predict travel times. Chen et al. (2004) use a neural network to predict bus travel times by aggregating states to their day of week, time of day, weather, and road segment. Chen et al. (2004) extent their offline approach with an online component. They refine estimations by processing real-time vehicle locations with the help of a Kalman filter. Lee (2009) predicts freeway travel times. The author aggregates the state space by k-means clustering and predicts travel times on the resulting clusters with the help of a neural network. Bertsimas et al. (2019) assign travel times to arcs in a network based on noisy origin-destination data. They add a regularization term to their cost function in order to generalize from missing data and introduce an iterative path generation scheme to reduce the underlying large-scale mixed-integer problem to a problem solvable in reasonable time. Huang et al. (2020) employ tree-based ensembles to predict the duration of taxi trips in New York based on a set of handcrafted features. They only consider trips between two locations and not entire routes.

Few works in time estimation for transportation and logistics employ online methods. Chu et al. (2005) predict freeway travel times based on real-time location data with the help of a Kalman filter. For our problem, the vehicle positions are known but if this was not the

case we could integrate a Kalman filter into our method in a similar fashion. Prokhorchuk et al. (2019) estimate travel time distributions for each arc of a road network by means of Bayesian inference trained on floating car data. Florio et al. (2020) propose a gradient-based method to assign travel times to road segments. The methods of Prokhorchuk et al. (2019) and Florio et al. (2020) can either be trained offline and then applied on new data or perform online updates of travel time on road segments based on current travel time observations. Online methods excel on simple problem structures where no difficult decisions have to be made during simulations. This is not the case in vehicle routing problems (VRPs) where simulating routing and assignment decisions is costly. In our online-offline method, we approximate a given routing policy with the help of a neural network. The approximate routing policy is then used to accelerate routing decisions in online simulations; enabling us to estimate arrival times by a sampling mean in real time. To the best of our knowledge, there is no comparable approach in the literature on time estimation for transportation and logistics. However, there are related ideas on learning routing and assignment decisions in the field of machine learning for static VRPs. We provide more details in the next subsection.

The RMDP poses a new challenge in time estimation. Due to its unique combination of uncertainties, none of the existing solution approaches are tailored to the RMDP. We fill the gap with an offline method that captures the unique features of the RMDP and with an online-offline method that enables detailed simulations of the RMDP in real time.

2.4 Machine Learning in Vehicle Routing

In this subsection, we embed our online-offline solution method in the literature. We learn a routing policy supervised with a neural network to accelerate simulations. Our neural network takes an existing route and a delivery request as input and predicts how a given routing policy would integrate the request into the route. The network is trained on a set of observed routing decisions made by the routing policy. We employ the network as a faster substitute of a given routing policy.

There are several related ideas in the traveling salesperson problem (TSP) literature, where an optimal but slow solution method is replaced by a faster neural network. Bello et al. (2016), Deudon et al. (2018), and Miki et al. (2018) encode TSP instances in a format

suitable for a neural network architecture to derive tours that solve the instances. The work of Miki et al. (2018) shares similarities with our idea. Miki et al. (2018) encode TSP instances as images to leverage the potential of convolutional neural networks (CNN). Using supervised learning, they interpolate between known optimal solutions of TSP instances. Though, for large scale instances, where no optimal solutions are known, they fall back on training their CNN using reinforcement learning.

In contrast to our supervised learning approach, Bello et al. (2016) and Deudon et al. (2018) focus on reinforcement learning. Bello et al. (2016) derive heuristic solutions to the TSP by mapping a set of city coordinates to a city permutation by means of reinforcement learning. They employ a recurrent network to encode city coordinates and a pointer network to point to iteratively built routes. This allows them to generalize their approach to solve TSPs of variable size. Deudon et al. (2018) use the same approach but choose a feed-forward neural network, instead.

In dynamic routing (like our problem), there is only work on reinforcement learning. A frequent solution approach is value function approximation (VFA). An approximate value function estimates the long term benefit of being in a state and therefore allows for far-sighted decision making in complex environments. Value functions are commonly approximated by means of reinforcement learning. The reader is referred to Ulmer (2020) for an overview of VFA in dynamic vehicle routing. However, approximating a value function by means of reinforcement learning is not suitable to estimate arrival times. Work on using neural networks (NN) in supervised learning for VRPs is particularly scarce. Mańdziuk (2018) provides an overview of recent advances in the VRP literature over the last few years, where the author points out that "it came as a surprise to the author that NNs have practically not been utilized in the VRP domain ...".

Work on online-offline methods is even more rare. An exception is the work by Ulmer et al. (2019) where they use an offline VFA in an online simulation to decide about acceptance or rejections of customers. The routing is conducted in full detail leading to calculation times of several minutes per customer request. To the best of our knowledge, we believe to be the first to approximate routing decisions offline in order to perform online simulations of a dynamic vehicle routing problem in real time.

3 Problem Definition

In this section, we define the problem of estimating arrival times for delivery routing (EAT). The problem is part of the restaurant meal delivery problem with arrival time estimation (RMDPEAT), an extension of the RMDP presented in Ulmer et al. (2020). In the RMDPEAT, customers order restaurant delivery over the course of the day. The orders are picked up and delivered by the platform's delivery fleet. Each customer's choice is informed by estimated arrival times for each restaurant. The goals of the RMDPEAT are twofold: serve many customers and provide accurate arrival time estimations. The latter is the goal of the EAT and the focus of this work. In the following, we first describe the dynamic decision process of the RMDPEAT and highlight the role of EAT. We then provide an example for a decision state of the RMDPEAT to prepare the subsequent definition of EAT.

3.1 Restaurant Meal Delivery Problem with Arrival Time Estimation

In the classical RMDP, customers spontaneously order meals from a list of restaurants through a restaurant-meal-delivery platform. Each meal is picked up by a driver at the restaurant and delivered to the respective customer. Drivers are permitted to bundle meals during their route before delivering them. We extend the classical RMDP by assuming that restaurant meal delivery services offer customers an estimated time of arrival for each restaurant. Given that information, customers choose from restaurants based on their preferences (in restaurants and delivery times). Thus, the customer may decide not to order at all. We illustrate the entire RMDPEAT process as a business processing modeling notation (BPMN) model in Fig. 1. We depict the different actors in the RMDPEAT as swimlanes. The interactions between actors are modeled as a flowchart. The process is started when a customer requests delivery (mail symbol); e.g. by visiting the website of the restaurant-meal-delivery platform. The customer is asked to enter their address data and the dispatcher checks information about the status of fleet and restaurants. Based on the information, the dispatcher estimates the arrival time at the customer for each potential restaurant and communicates the estimated arrival time to the customer. Informed by the arrival times, the customer

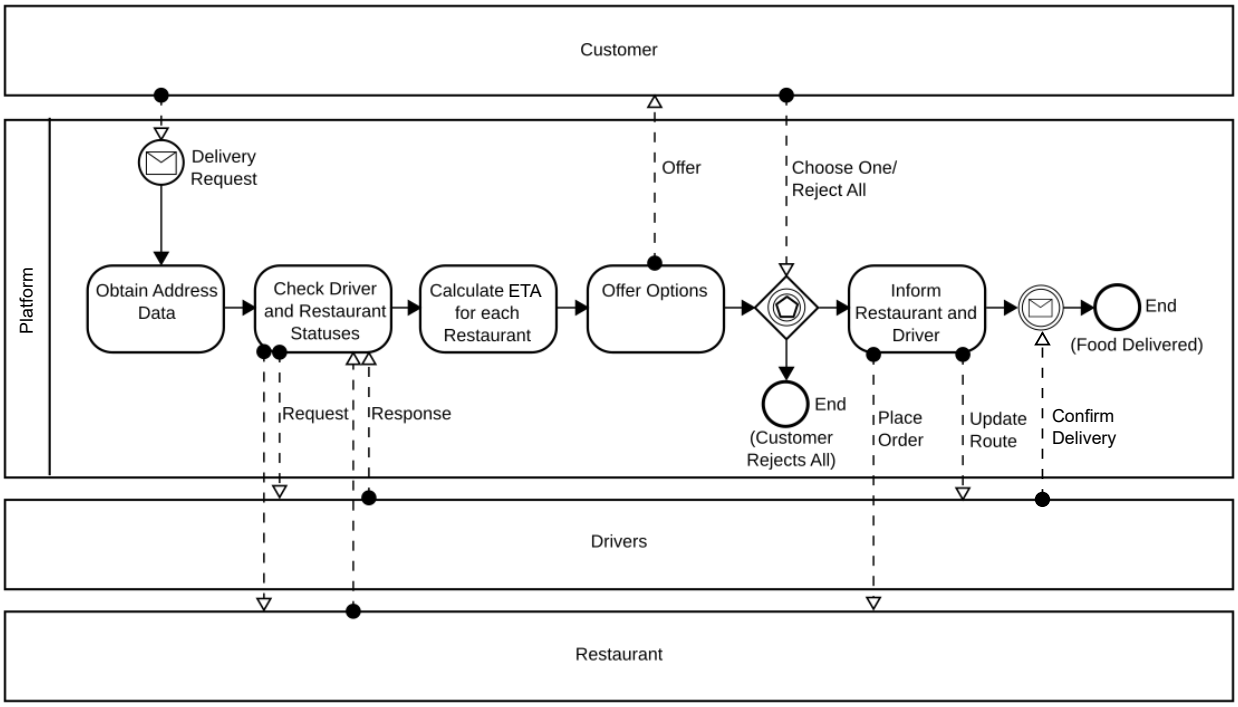

Figure 1 BPMN-Model of Process from Customer Order to Meal Delivery

chooses a restaurant for delivery or decides not to order at all. If the customer does not order, the process ends. If the customer chooses a restaurant for delivery, the dispatcher informs the corresponding restaurant and updates the routes for the drivers. In the latter case, the process ends when the meal arrives at the customer.

Formally, we define the RMDPEAT as a dynamic decision process. This modeling framework depicts dynamic problems as a sequence of states, decisions, and new information (Powell 2011).

Decision Points: A decision point k occurs whenever a customer requests estimated arrival times (see Fig. 1, "Delivery request").

States: The state S_k at decision point k consists of the time of the request t_k, the requesting customer c_k, all customers $\mathcal{N}_k$ that have ordered but were not served yet, the restaurants and their workloads $\mathcal{R}_k$, and the set of planned routes Θ_k. Each restaurant is given by a location and a queue of meals to prepare and the start of preparation time of the first item in the queue. Each route $\theta \in \Theta_k$ is a sequence of pickup and delivery stops, called actions. Actions are given by an origin, a destination, the time when the

action is started or is planned to be started, and an assumed time required to perform the action.

Decisions: Decisions are made in two stages. First, we decide on tentative route updates for each restaurant in case the customer orders from the restaurant. Second, we estimate arrival times for each restaurant based on the current state and the tentative routes (see Fig. 1, "Calculate ETA for each Restaurant"). We denote the tentative routes as Θ_{ik} and the arrival time decisions as $X_{ik} := X(S_k \cup \Theta_{ik}) \in \mathbb{R}$ for every restaurant $i = 1, \ldots, |\mathcal{R}_k|$. The estimated arrival times $\big(X_{ik}\big)_{i \in \{1,\ldots,|\mathcal{R}_k\}}$ are communicated to the customer (see Fig. 1, "Offer Options"). Selecting the estimated arrival times is the focus of this work and is described in detail in the next subsection.

Stochastic Information: We consider four sources of uncertainty: First, customers are unknown until they request delivery and estimated arrival times. Second, meal preparation times at restaurants are uncertain. Third, parking and delivery times of drivers are uncertain. The fourth source of uncertainty is the restaurant choice of customer c_k. Customers decide on the basis of individual preference functions $\Phi_k\big((X_{ik})_{i \in \{1,\ldots,R\}}\big)$. The preference function takes a vector of estimated arrival times (one for each restaurant) as input and returns a restaurant to order from or rejection of service.

Stochastic Transitions: The stochastic transition from state S_k to state S_{k+1} is induced when a customer requests estimated arrival times. The new set of planned routes Θ_{k+1} is the result of our routing decision and the restaurant choice of customer c_k at time t_k. The new routes are updated by pruning actions that were completed between t_k and t_{k+1}. The completion of an action depends on the realizations of stochastic parking and meal preparation times. The restaurant workloads $\mathcal{R}_{k+1}$ depend on the customer's choice and the realizations of meal preparation times. The set of customers yet to be served $\mathcal{N}_{k+1}$, depends on the four types of information and the routing decision.

Objective: The objectives of the RMDPEAT are twofold: Maximize the expected number of served customers and estimate arrival times accurately. The accuracy of arrival time estimations is measured by the expected mean squared error (MSE) per customer.

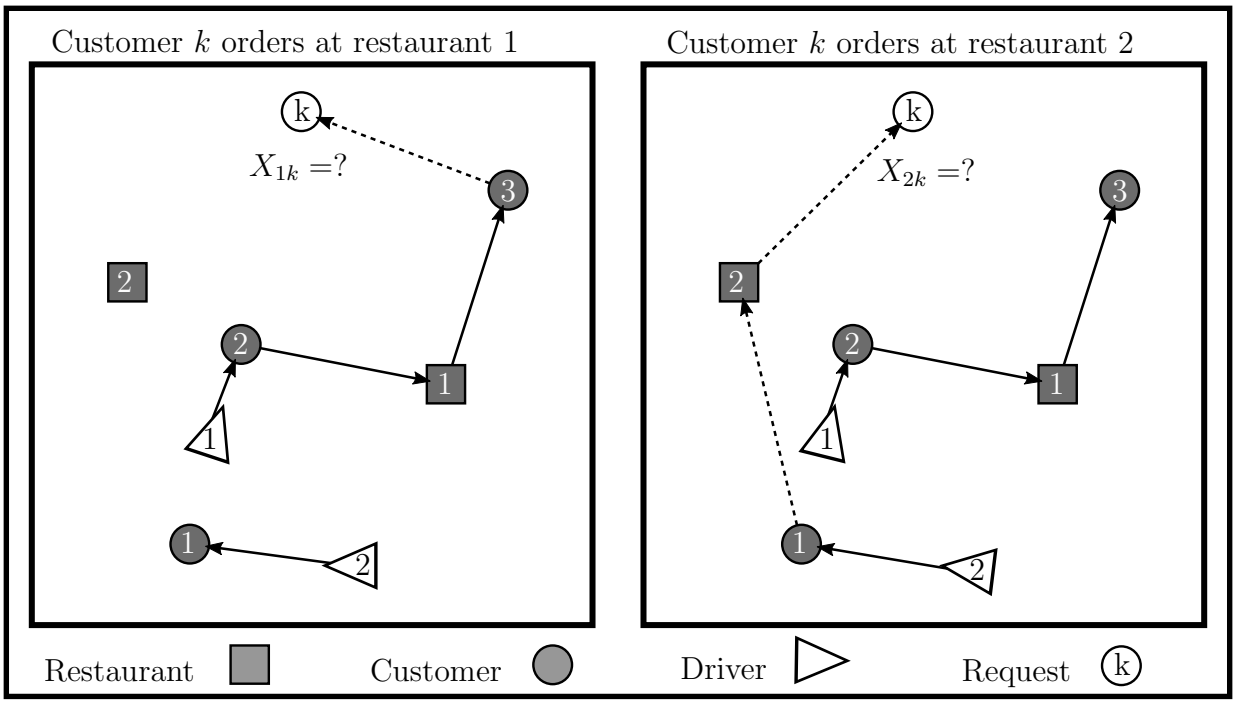

Figure 2 State S_k, Routing Decisions $(\Theta_{1k}, \Theta_{2k})$, and EAT decision (X_{1k}, X_{2k}).

In this paper, we focus on the EAT in the RMPDEAT. We therefore assume the platform routes customers using an insertion procedure - a method commonly applied in practice. We build routes by iterating over all existing tours and all possible points of insertion for pickup and delivery such that the sum of tour durations is minimized. We allow the bundling of orders at same or adjacent restaurants; i.e. a driver is allowed to wait for more than one meal before delivering them (See Appendix A.1 for results without bundling). To avoid substantial shifts of customers in a route, we restrict the routing policy so that customers cannot be shifted by more than 15 minutes in total due to insertions.

3.2 Estimating Arrival Times for Delivery Routing Problem

When a customer requests delivery options, a decision state for the EAT occurs. We illustrate an exemplary decision state of the EAT in Fig. 2. Light triangles represent the drivers, gray squares the restaurants $\mathcal{R}_k$, gray circles the previously assigned customers $\mathcal{N}_k$ that have not been served yet, and the light circle is a new request by the customer that induced the kth decision state. The black arrows indicate the current routes and the dashed arrows indicate

tentative route updates to serve customer k. For the purpose of presentation, we omit the depiction of assumed action times.

In this example with two restaurants, two estimated arrival times must be communicated to the customer. Therefore, we have to consider two scenarios. In the left scenario, customer k orders at restaurant 1. Here, driver 1 delivers a meal to customer 2, picks up the meals of customer 3 and k at restaurant 1, serves customer 3 and then customer k. In the right scenario, customer k orders at restaurant 2. Here, driver 2 delivers a meal to customer 1, picks up the meal at restaurant 2 and delivers it to customer k. This decision of the EAT is based on the state S_k and the two tentative planned routes Θ_{1k}, Θ_{2k}. The decision $\big(X_{1k}, X_{2k}\big)$ is subsequently communicated to the customer, who either decides on a restaurant for delivery or refrains from ordering at all. In case the customer selects a restaurant, the tentative planned route is implemented and followed until the next customer requests delivery.

As illustrated in the example, a state in EAT consists of a RMDPEAT state combined with the tentative routing decision for each restaurant. In the following, we define the decision process and the objective of EAT.

Let c_k be the customer that induced the kth decision state. The corresponding RMDPEAT decision state is given by $S_k = \big(t_k, \mathcal{N}_k \cup \{c_k\}, \mathcal{R}_k, \Theta_k\big)$. As customer c_k has not ordered yet, there are $|\mathcal{R}_k| + 1$ potential routing decisions, one for each restaurant and one for the option not to order. A decision state S_k^{EAT} in EAT is then given by the RMDPEAT state S_k combined with routing decisions Θ_{ik} for all restaurants $i = 1, \dots, |\mathcal{R}_k|$.

A decision in the EAT is a vector of estimated arrival times $X(S_k^{\text{EAT}}) := (X_{ik})_{i \in \{1,\dots,|\mathcal{R}_k|\}}$, where each entry corresponds to a restaurant. The objective is to minimize the expected deviance of the estimated arrival time $X(S_k^{\text{EAT}})(j)$ from the actual arrival time $A(S_k^{\text{EAT}})(j)$

$$\min_{X(S_k^{\text{EAT}})(j)) \in \mathbb{R}_+} \mathbb{E}_{S_k^{\text{EAT}}}\big(||A(S_k^{\text{EAT}})(j) - X(S_k^{\text{EAT}})(j)||_2^2\big), \tag{1}$$

where $j \in \{1, \dots, \mathcal{R}_k\}$ is the restaurant that the customer will select. We choose the mean squared error (MSE) to describe the deviance of estimated arrival times from actual arrival times.

Evidently, there are interdependencies between the objectives of the RMDPEAT. Sophistically, one could sacrifice one to optimize the other, for example, by communicating arrival times of zero (many services, large arrival time deviations) or by adding large amounts of

slack to the routing (few services, accurate arrival time predictions). While these interrelations may be worth investigating in the future, we focus on the case where the routing policy is given and our goal is to minimize the deviation of communicated and realized arrival time.

4 Solution Approach

In the following, we present our solution approaches for EAT. We present an offline method and an online-offline method. We motivate both methods in Section 4.1 and provide detailed explanations in Sections 4.2 and 4.3.

4.1 Motivation

In theory, an optimal policy for EAT is achieved by selecting the expected arrival time $\mathbb{E}\big(A(S_k^{\mathrm{EAT}})\big)$ given the state S_k^{EAT}. In practice, we cannot analytically determine the expected arrival time due to uncertainty in delivery times, due to uncertainty in bundling, and due to the interaction of the delivery and meal preparation process. When designing solution methods for EAT, we have two conflicting objectives in mind. On the one hand, solutions have to be fast to predict arrival times for multiple restaurants in real time. On the other hand, solutions are required to yield accurate predictions.

One way of a near-optimal approximation of the unknown distribution of arrival times are full *online* simulations of delivery and preparation process, future demand, and routing. For such a process, the platform requires a simulator and, as we show later, substantial runtimes per customer. We describe the procedure for a single simulation in Alg. 1. In the procedure, we already introduce two simplifications to accelerate the simulation, both concerning the handling of new customer requests within the simulation. First, we do not estimate the arrival time for every restaurant before letting a simulated customer decide based on all arrival time estimations. Instead, we iteratively communicate one estimated time of arrival to the simulated customer. This does not change exactness of the simulation if the assumption holds that the customer choice model can be modeled as a sequence of yes or no decisions. Second, we use a myopic heuristic described in the next paragraph to estimate arrival times in the simulation instead of calling Alg. 1 recursively. The procedure

Algorithm 1 One Full Simulation

```
1: Given current state (current time, new (real) customer, waiting customers, restaurants
   and their workloads, current routes), assignment and routing policy
2: sample request stream (sample customers and sort by order times)
3: while new (real) customer is not served do
4:     for customer in request stream do
5:         transition to the next state
6:         randomly permute the restaurants
7:         for restaurant in permuted restaurants do
8:             construct tentative route using assignment and routing policy
9:             Estimate arrival time based on tentative route
10:            communicate estimated arrival time to customer
11:            if customer places an order (based on choice model) then
12:                update routing (based on tentative route)
13:                break For-Loop
14: return arrival time at new (real) customer
```

takes the current state as input (line 1) and simulates the arrival time of the new (real) customer as follows: The future demand is sampled from the known temporal and spatial distribution of requests and stored in a list sorted by order times (line 2). As long as the current customer is not served (line 3), we iterate through each sampled customer in the request stream (line 4). We do so by transitioning from the current state to the state at the time of the request (line 5). We permute the restaurant list to model preferences of the customer. For each restaurant (line 6) in the permuted list, we construct a tentative route using a given routing and assignment policy to deliver a meal from the restaurant to the requesting customer (line 7). We derive an estimated arrival time based on the tentative route (line 8) using the myopic heuristic described in the next paragraph. Informed by the estimated time of arrival (line 9), the requesting customer either chooses to order based on their choice model (line 10) or checks the next restaurant. We update the routing (line 11) using the previously determined tentative route and stop iterating over the restaurants if

the customer places an order. The procedure ends when the new (real) customer is served. Upon completion, Alg. 1 returns the simulated arrival time at the new customer (line 12).

Leaving out our proposed simplifications, applying the procedure multiple times results in a sampling mean that is an unbiased estimator of the expected arrival time. With increasing numbers of simulations the sampling mean converges to the expected arrival time almost with certainty by the strong law of large numbers. We note that our simulation is not exact as we do not estimate arrival times during the simulation using another simulation (line 8). As a result, our full online simulation has a small bias. This cannot be avoided without accepting the computationally intractable runtimes of recursive simulations.
However, the runtime decrease achieved by our simplifications is a mere drop in the ocean. Full online simulations are still very time-expensive and therefore infeasible when arrival times have to be computed on the fly. In particular, routing and assignment decisions (line 9) are runtime-expensive for a combinatorial problem like the one at hand. In our experiments, we use an insertion procedure that is often applied in practice. Even such a simple policy scales quadratic with regards to the number of actions in a route and linear with regards to the number of vehicles in the fleet, thus cubic. In a worst-case scenario, this heuristic is applied for every restaurant and every simulated customer request (line 4 and 6) during a simulation run.

An alternative way of estimating the expected arrival times is by a simple sum of expected times for each action in the route. We call this approach planning on means (PoM). By planning on mean action times we assume the underlying RMDPEAT to be deterministic instead of dynamic. Such a myopic approach is bound to yield inaccurate results when the assumption of a deterministic RMDPEAT diverges from the dynamicity of the real RMDPEAT. In particular, the uncertainty in bundling of orders is not addressed by PoM. We illustrate this issue in Fig. 3. At $t = 813$ (1:33 pm), the driver intends to deliver a meal to customer 1, to drive to the next restaurant, to pick up the food for customer 2, and to deliver it directly. Planning on means assigns every action in the planned route a time estimate. The arrival time estimations for customer 1 and 2 are then given by the respective sum of estimated times. At $t = 841$ (2:01 pm), we observe the actual arrival times. While the time estimate for customer 1 was only off by one minute, it was off by eight minutes for

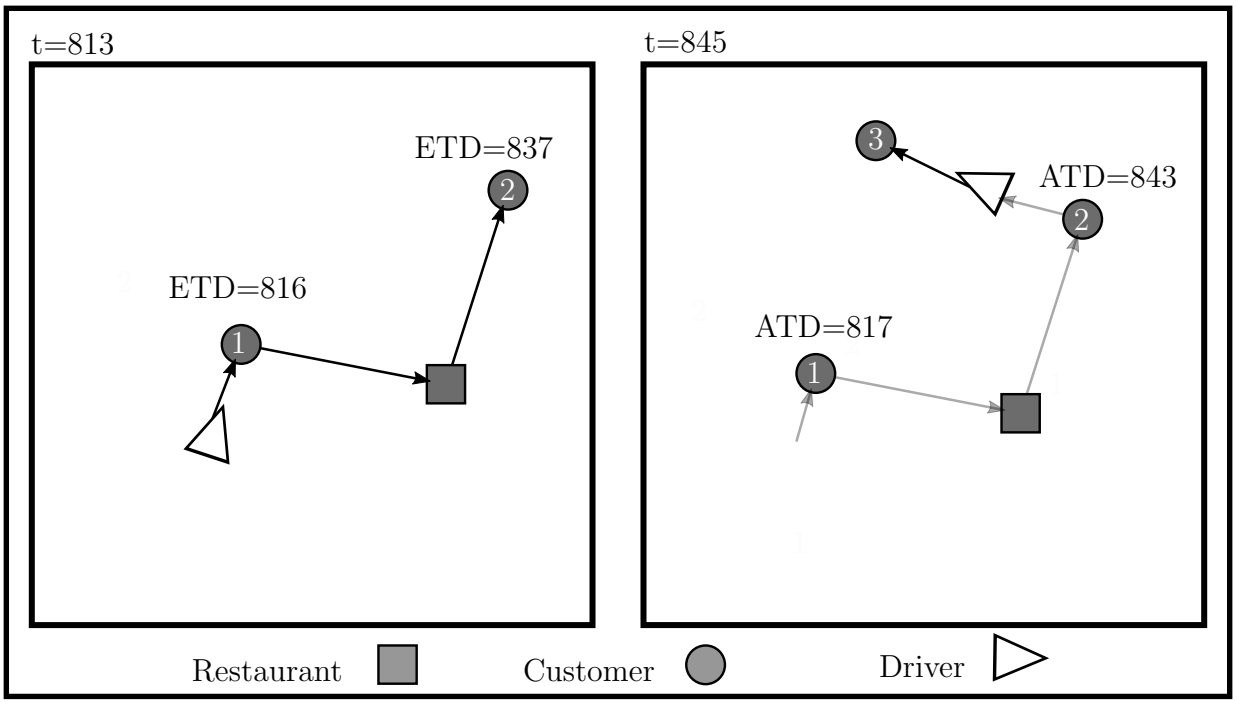

Figure 3 Exemplary Arrival Time Estimation by Planning on Means

customer 2. This is due to the request of customer 3 who ordered at the same restaurant as customer 2. Instead of delivering the meal directly to customer 2 as assumed at $t = 813$ (1:33 pm), the driver waited for both meals to be prepared before delivering them.

PoM underestimates arrival times even when no bundling occurs. This is due to the fact that PoM ignores the interaction of delivery and preparation process when estimating departure times at restaurants:

The expected departure time $\mathbb{E}\big(t_{\text{dep}}\big)$ is given by the expected value of the maximum of the pickup time t_{pickup}, which is the sum of travel time to the restaurant and the time required to find a parking space, and the time when the meals are ready t_{cook}

$$\mathbb{E}\big(t_{\text{dep}}\big) = \mathbb{E}\big(\max\{t_{\text{pickup}}, t_{\text{cook}}\}\big). \tag{2}$$

As PoM assumes deterministic information, it approximates the expected departure time by

$$\max\{\mathbb{E}\big(t_{\text{pickup}}\big), \mathbb{E}\big(t_{\text{cook}}\big)\}. \tag{3}$$

PoM is biased because approximation (3) is likely to underestimates the departure time, as we show in the following.

Lemma 1. *Let X, Y be random variables. Then,*

$$\mathbb{E}[\max(X, Y)] \geq \max(\mathbb{E}[X], \mathbb{E}[Y])$$

Proof. We use the identity

$$\max(X, Y) = \frac{X + Y + |X - Y|}{2} \tag{4}$$

and the inequality

$$\mathbb{E}[|X|] \geq |\mathbb{E}[X]| \tag{5}$$

as well as the linearity of the expected value to show the claim:

$$\begin{aligned}
\mathbb{E}[\max(X, Y)] &\overset{(4)}{=} \mathbb{E}\left[\frac{X + Y + |X - Y|}{2}\right] \\
&\overset{(lin.)}{=} \frac{\mathbb{E}[X] + \mathbb{E}[Y] + \mathbb{E}[|X - Y|]}{2} \\
&\overset{(5)}{\geq} \frac{\mathbb{E}[X] + \mathbb{E}[Y] + |\mathbb{E}[X - Y]|}{2} \\
&\overset{(lin.)}{=} \frac{\mathbb{E}[X] + \mathbb{E}[Y] + |\mathbb{E}[X] - \mathbb{E}[Y]|}{2} \\
&\overset{(4)}{=} \max\big(\mathbb{E}[X], \mathbb{E}[Y]\big).
\end{aligned} \tag{6}$$

□

Corollary 1.

$$\mathbb{E}\big(t_{dep}\big) \geq \max\{\mathbb{E}\big(t_{pickup}\big), \mathbb{E}\big(t_{cook}\big)\}. \tag{7}$$

Proof. The inequality follows directly from Lemma 1. □

In summary, full online simulations and PoM are two extremes in terms of the accuracy-runtime trade-off. While full online simulations are highly accurate, they are too slow for application. In contrast, PoM predicts arrival times instantly but inaccurately. In the following, we propose two methods, a pure offline and an online-offline method, that alleviate the shortcomings of PoM and full online simulations while preserving their advantages. The offline method addresses the short-sightedness of planning on means while retaining its speed. It can be applied whenever historical data is available and does not require a simulator at all. It learns the residuals of PoM, i.e. the difference of estimated arrival times and actual

arrival times, by means of gradient boosted decision trees (GBDTs). For that purpose, we derive a low-dimensional feature space that describes key information of the EAT decision space. The features account for all sources of uncertainty and enable the offline method to compensate for the bias of planning on means.

We also present an online-offline method that can be integrated in the simulator potentially already employed by the platform. The online-offline method drastically decreases the computation time of full online simulations by approximating the routing policy while preserving their accuracy. Our online-offline method learns the decision making of the routing and assignment policy offline with the help of a deep neural network (DNN). We employ the trained DNN online to perform simulations in a fraction of the time previously required.

4.2 Offline Method

As we have seen, planning on means ignores the uncertainty in bundling and the uncertainty in synchronisation of delivery and meal preparation process. We implement a method to account for the short-sightedness of PoM by approximating its residuals with the help of supervised learning. To address the high dimensionality of the state space, we use a set of features that encode the state space in a suitable format for our supervised learning model. In the following, we describe approximation procedure and feature selection.

4.2.1 Approximation Procedure

When predicting residuals, we decide for a classical machine learning approach over a deep learning approach. This way, we decrease training time, which will be of importance for our extensive analysis in Section 6, and ensure interpretability of our model with regards to the handcrafted features (Du et al. 2019). We choose a gradient boosted decision tree (GBDT) model for predicting the residuals. GBDT is a common method in the machine learning community that has shown success in various machine learning challenges (Bissacco et al. 2007, Hutchinson et al. 2011, Pittman and Brown 2011, Wang et al. 2017). GBDT is an additive ensemble of decision trees, where each tree partitions the feature space into disjoint regions and assigns a constant value to each region. A single decision tree is prone to overtraining. GBDT alleviates this issue by summing up several small trees. The decision

trees are trained sequentially using steepest descent by fitting the residual error of the current model. In the following, we give a short introduction into GBDT. For a detailed description, the reader is referred to Friedman (2001). Let $\mathbf{x}$ be a point in the feature space. A single decision tree is given by

$$f(\mathbf{x}) = \sum_{j=1}^{J} b_j \mathbf{1}_{\{\mathbf{x} \in R_j\}}, \tag{8}$$

where J is the number of terminal nodes in the tree and $\{R_j\}_{j=1}^{J}$ are disjoint regions that cover the feature space. $\{b_j\}_{j=1}^{J}$ are the constant values corresponding to the regions R_j. The additive ensemble is sequentially trained by adding a tree in every step:

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \rho_m \sum_{j=1}^{J} b_{jm} \mathbf{1}_{\{\mathbf{x} \in R_{jm}\}}. \tag{9}$$

Here, the index m denotes the tree from the mth iteration and F_m consists of m trees. The scaling factor ρ_m is the result of a line search using steepest descent. Let y denote the residual of planning on means that corresponds to $\mathbf{x}$ and let $L\big(y, F(\mathbf{X})\big)$ be a loss function. Then, the direction of steepest descent for the data pair $(\mathbf{x}, y)$ is given by

$$-g_m(\mathbf{x}) = -\left[\frac{\partial L\big(y, F_m(\mathbf{x})\big)}{\partial F_m(\mathbf{x})}\right]. \tag{10}$$

Finally, ρ_m can be derived from the line search

$$\rho_m = \arg\min_{\rho} \mathbb{E}_{x,y} L\big(y, F_{m-1}(\mathbf{x}) - \rho g_m(\mathbf{x})\big). \tag{11}$$

Training decision trees is time consuming. Finding the best split points in the feature space is a complex task that grows more difficult when the number of training samples or the number of considered features increase. The *lightGBM* implementation of GBDT models (Ke et al. 2017) addresses this limitation. It effectively reduces training samples by keeping those with large gradients and randomly dropping samples with low gradients. Additionally, Ke et al. (2017) introduces an almost lossless approach to bundle features, thus reducing the feature space. We employ the *lightGBM*-GBDT model for our computational study.

4.2.2 Feature Selection

We select features with two conflicting objectives in mind: We want to minimize the prediction error of our supervised learning model while keeping the dimension of our feature space

small to avoid the curse of dimensionality. Additionally, we want features to be robust with respect to changes in the fleet size or in the number of restaurants. This way, the model can be employed even when the fleet size or the set of restaurants changes. In the following, we present all features considered. We group them into four classes: temporal features, spatial features, routing features, and process features. We intend the features to enable the model to anticipate future bundling and delay due to synchronisation of delivery process and meal preparation process.

Temporal Features. We propose two temporal features that account for the temporal uncertainty in customer requests. The features

- *order_time*: Order time (daytime in minutes)
- *eta_pom*: Arrival time estimated by PoM (daytime in minutes)

lay the time frame in which bundling might occur. We expect our method to shift an arrival time estimated by PoM to a later time if the time frame is during peak lunch or dinner time. The arrival time estimated by PoM also gives an estimate on the length of the tour to the customer.

Spatial Features. The spatial locations of customer and restaurant influence arrival times. If customer and restaurant location span an axis along areas with ample potential customers and restaurants future bundling might be more likely. As a consequence, our method might correct the arrival time estimated by PoM to a slightly later time. For that reason, we propose two spatial features that address capture spatial information:

- *customer_location*: Customer location (block index of a 64×64 grid)
- *restaurant_location*: Restaurant location (block index of a 64×64 grid)

We fit a 64×64 grid onto the service area to represent two-dimensional locations. A location is encoded as an integer block index $i \in \{0, \ldots, 4095\}$ of the fitted grid.

Route Features. The length and structure of the route substantially shapes the likelihood of future bundling, the magnitude of uncertainty in parking and preparation, and delays due to process synchronisations. We propose five route features. The first three

features describe the structure of the route. The last two features account for the shift-constraint of our insertion policy. The constraint asserts that a delivery to a customer must not be delayed by more than 15 minutes due to bundling. Information on previous shifts due to bundling enables the model to better estimate the likelihood of future delays due to rerouting that affect the new customer.

- *n_stops*: Number of stops in the route
- *n_pickup_stops*: Number of pickup stops in the route before the customer is served
- *n_delivery_stops*: Number of delivery stops before the customer is served
- *max_shift_before*: Maximal time shift due to past insertions/bundling (in minutes) over all customers in the route that are served before the new customer
- *max_shift_after*: Maximal time shift due to past insertions/bundling (in minutes) over all customers in the route that are served after the new customer

Process Features. Finally, we use a single feature to address the uncertainty in processing times:

- *prep_time*: Estimated time until the meal is prepared by the restaurant based on expected preparation times (in minutes)

With the help of the estimated process time and the route features, the method is able to refine the approximation bias (see Corollary 1) when estimating wait times at restaurants.

We train and employ the GBDT using above features. The method can be trained exclusively on historical data and does not require a simulator to obtain training data. It is therefore instantly applicable for platforms that do not employ simulators.

4.3 Combination of Offline and Online Methods

In the previous section, we derived an offline method that is fast but uses aggregated data. More detailed estimations are obtained by full online simulations (see Alg. 1). However, full online simulations are computational infeasible when arrival times have to be computed

in real time for multiple restaurants. We exploit an offline approximation of the routing policy to derive an online simulation that estimates arrival times on the fly. This is achieved by training a DNN to imitate a given routing policy. The new approximate routing policy replaces the exact routing policy in line 7 of Alg. 1. Furthermore, we remove the for-loop in line 6 by assuming that customers choose the restaurant for delivery uninformed and uniformly random.

In the following, we describe our approach. As for the definition of the offline approach, we divide this subsection into approximation procedure and feature selection.

4.3.1 Approximation Procedure

We present the architecture of our DNN before we provide details on our approximate online simulation. The DNN requires information on the customer, the restaurant, and the vehicle. It returns two output vectors that predict the insertion point for restaurant as well as customer into the vehicle's route. Each output vector is computed by a head of fully connected layers. The output represents a likelihood distribution over all insertion points. The customer's insertion point depends on the insertion point of the restaurant, because the customer is served after the pickup at the restaurant. For this reason, we feed the likelihood distribution over the restaurant insertion points back into the DNN to guide it determine the customer insertion point. The entire model is depicted in Fig. 4. In the following, we describe each step in the DNN model. Steps 1 to 6 describe a forward pass applying the DNN while steps 7 to 10 describe a backward pass training the DNN.

1. The normalized input vector v of length N is forwarded to the input layer. The vector is later described in the feature section.

2. The input layer consists of a single linear layer with a ReLu activation.

3. The resulting vector is forwarded to the first head of fully connected layers. We use four layers with 128 nodes each. All layers are followed by a batch normalization, and a ReLu activation. The layers are concluded by a linear layer of $r(N)$ nodes. Here, $r(N)$ denotes the number of possible insertion points for the restaurant.

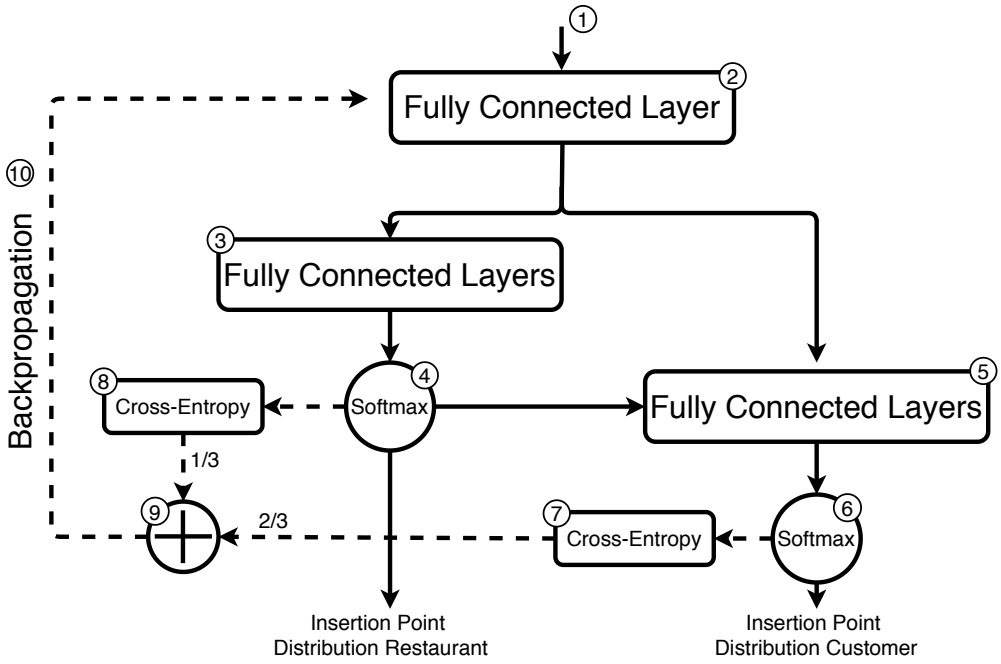

Figure 4 DNN Routing Model

4. The $r(N)$ dimensional output vector is forwarded to a softmax activation. The resulting output is a likelihood distribution over the $r(N)$ potential insertion points. The index with maximal likelihood is used as the insertion point for the restaurant.

5. The route embedding from step 4 and the output vector from step 6 are concatenated and forwarded to the second head of fully connected layers because the customer can only be inserted after the restaurant. We use the same structure as in the first head of fully connected layers. The only difference is a linear output layer of size $c(N)$. This time, $c(N)$ denotes the number of possible insertion points for the customer.

6. The $c(N)$ dimensional output vector is forwarded to a softmax activation. The resulting output is a likelihood distribution over the $c(N)$ potential insertion points. The index with maximal likelihood is used as the insertion point for the customer.

7. The cross-entropy loss between the likelihood distribution for the customer insertion point and the true customer insertion point is computed.

8. The cross-entropy loss between the likelihood distribution for the restaurant insertion point and the true restaurant insertion point is computed.

9. A linear combination of both losses is computed. We weight the customer loss by 2/3 and the restaurant loss by 1/3. This is done for practical reasons as the customer loss is harder to learn.

10. The resulting loss is backpropagated through the entire neural network.

We use the Adam optimizer (Kingma and Ba 2014) to train the DNN. Adam is the default optimization method in many deep learning applications. In contrast to classical stochastic gradient-descent, which uses fixed learning rates, it computes adaptive learning rates for each parameter based on the first and second moments of the gradient. We use a cosine annealing learning rate strategy with warm restarts (Loshchilov and Hutter 2016). Warm restarts are known to improve rate of convergence in gradient-based optimization. Within each iteration the learning rate is decayed by cosine annealing to enforce convergence after the warm restart.

With the help of the trained DNN, a routing decision for a given vehicle can be performed in $\mathcal{O}(C)$ time, in contrast to the $\mathcal{O}(|\theta|^2)$ time required by our simple insertion policy. Here, C denotes the number of elementary operations required for one forward pass of the neural network and $|\theta|$ denotes the number of actions in the route. The approximate routing and assignment policy is described in Alg. 2 and is summarized as follows: For each route, we predict the insertion points of customer and restaurant with the help of the DNN. If an insertion point is infeasible, restaurant and customer are appended to the end of the respective route. Else, we insert pickup and delivery according to the prediction. We choose a route from the resulting tentative routes based on the given assignment policy. Using the approximate routing and assignment policy we perform online-offline simulations as detailed in Alg. 3. The procedure is summarized as follows: When a customer requests delivery, we simulate the delivery process until the customer is served. We sample new requests according to the temporal and spatial distribution of orders. Each sampled customer requests delivery from a restaurant chosen uniformly random. We update the routing according to our approximate routing and assignment policy (see Alg. 2). When the new customer is served, we store the simulated arrival time. Multiple simulations yield an empirical distribution of arrival times. We estimate the arrival time by the mean of the empirical distribution. Though, we could

Algorithm 2 Approximate Routing and Assignment Policy

Given current state, customer, restaurant, assignment policy
tentative routes = $\emptyset$
for route in routes **do**
    predict insertion point for pickup and delivery using DNN
    if prediction is feasible **then**
        insert pickup and delivery according to prediction
    else
        append pickup and delivery to the end of the route
    tentative routes $\leftarrow$ tentative routes $\cup$ route
choose route from tentative routes with the help of the assignment policy
return route

Algorithm 3 One Approximate Simulation

Given current state (current time, requesting customer, waiting customers, restaurants and their workloads, current routes)
sample request stream (customers and sorted order times)
while current customer is not served **do**
    for customer in request stream **do**
        update state
        place an order at the given restaurant
        update routing according to approx. routing and assignment policy
return Arrival time at new customer

also derive confidence bands on the arrival time, for example, to communicate a delivery time *window* to the customer.

We note that the approximate routing policy introduces further bias into our estimator. Whenever the DNN predicts an infeasible insertion point, we append restaurant and customer at the end of the respective route. Therefore, we underestimate the likelihood of bundling. Consequently, the predicted arrival time is likely to underestimates the actual arrival time. Our computational study shows that this bias is smaller than one minute.

4.3.2 Feature Selection

We assume that each route consists of a maximum of N actions. We choose N large enough such that no route can have more than N actions. Each action describes either a pickup at a restaurant or a delivery to a customer. We represent an action by a five-dimensional vector.

- Action type (Pickup or delivery),
- Location of action (x- and y-coordinate),
- Estimated time required to perform the, action (in minutes)
- Total time that the customer was delayed by insertion policy (0 if pickup action).

If the route has less than N actions, we pad the input vector with zeros.
In addition to the route, the DNN is given

- Customer position (x- and y-coordinate),
- Restaurant position (x- and y-coordinate),
- Estimated time until the meal is prepared (in minutes).

In total, the neural network receives $5 \cdot (N + 1)$ input parameters. The maximal possible length of a route N depends on various factors. We observe that $N = 12$ is sufficiently large for our computational study, described in the next section.

5 Computational Study

In the following, we present our computational study. Experiments were conducted on the *North-German Supercomputing Alliance HLRN-IV-System Lise* at *Zuse Institute Berlin*. We used one node equipped with 364 GB working memory that consists of two CPUs with 48 cores each. Training the GBDT requires only a few minutes and training our DNN takes less than a day.

This section is structured as follows: Section 5.1 describes the experimental design. Section 5.2 explains the parametrization of our models. Finally, Section 5.3 compares the quality of all proposed EAT policies.

5.1 Design of Experiments

In this section, we describe the experimental design that forms the basis of our computational study. First, we present the spatial and temporal distribution of requesting customers and their ordering behavior. Second, we define the restaurant locations and the distribution of times required to prepare meals. Third, we describe the fleet of delivery drivers, their travel times, and their distribution of parking times.

5.1.1 Customers

The customer locations are taken from the RMDP data set `http://ir.uiowa.edu/tippie_pubs/69`. The data set includes 32853 unique customer locations in Iowa City. We assume that customers spontaneously request meal deliveries throughout the day (0 minutes to 1440 minutes). We set the expected number of requests to 450. The requests are split into lunch time requests and dinner time requests. We model the lunch time distribution as $\mathcal{N}(\mu = 720, \sigma = 60)$ and the dinner time distribution as $\mathcal{N}(\mu = 1080, \sigma = 60)$. Following the results of Dai and Liu (2019), we model the dinner demand slightly higher than the lunch demand. Each day, we sample $n^{\text{lunch}} \sim \mathcal{N}(\mu = 200, \sigma = 10)$ times from the lunch time distribution and $n^{\text{dinner}} \sim \mathcal{N}(\mu = 250, \sigma = 10)$ times from the dinner time distribution. The resulting bimodal distribution of delivery requests over time is illustrated in Fig. 5 (left). The x-axis shows the daytime in minutes and the y-axis the relative frequency of requests. In our experiments, we discretize request times to steps of one minute. For each request, we determine the estimated time of arrival for every given restaurant. Informed by the estimated arrival times, the requesting customer decides for a restaurant and places an order. We define the customer choice model Φ_k as follows: We assume that customers have an ordered set of potential restaurants to order from as well as a maximum time they are willing to wait for delivery. Thus, we model their choices by a list of restaurants ordered by the customer's preference and an arrival time threshold. Each customer iterates through their list up to the first restaurant where the ETA is below the threshold. Else, the customer rejects delivery service. In our main experiments, we assume that all restaurants are part of each list and that all customers have a delivery threshold of 60 minutes after time of order. We also assume that the positioning in the list is uniformly distributed among the

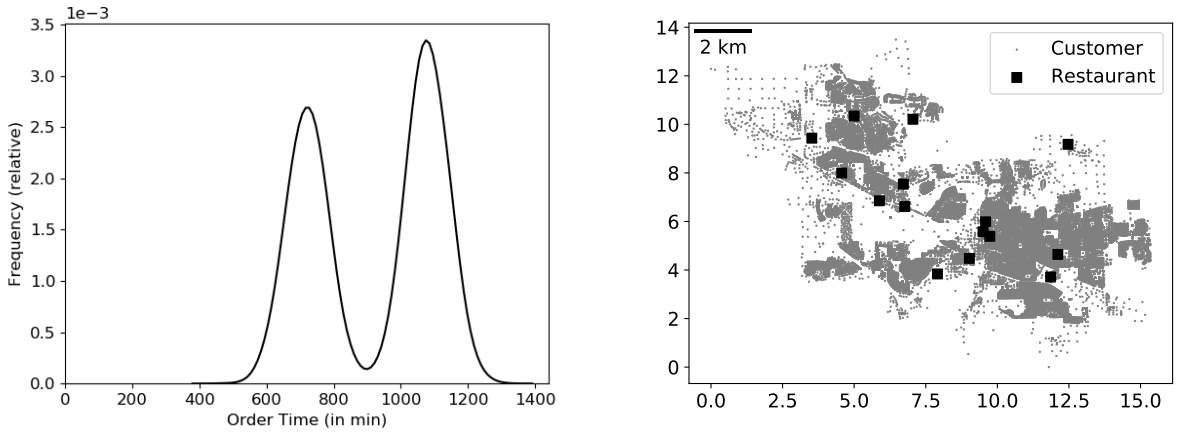

Figure 5 Temporal (left) and Spatial (right) Distribution of Requests

restaurants. This leads to customers choosing uniformly random from all restaurants with an estimated arrival time of less than one hour. In our analysis in Section 6, we later vary both list lengths and threshold to model more and less selective customer behavior.

5.1.2 Restaurant

We consider 15 restaurants in our computational study. The restaurants are representatives of the 110 restaurants present in the data set chosen by means of a 15-medoid clustering. The spatial distribution of restaurants and customers in Iowa City is illustrated in Fig. 5 (right). The x- and y-axis denote the coordinates of locations in kilometers. Grey dots depict the location of customer requests and black square indicate restaurant locations. Restaurants are characterized by a location and a sequence of meals to prepare. The time required to prepare a meal is modeled equally for all restaurants by a log-normal distribution $\mathcal{LN}(\mu = 8, \sigma = 1.3)$. We assume that restaurants prepare meals sequentially and in first-in-first-out fashion. We only know if a meal is ready when the assigned driver arrives at the restaurant.

5.1.3 Fleet

We assume a homogeneous fleet of drivers with unlimited capacity. In practice, drivers serve between two and three customers per hour. We have a time frame of approximately

ten hours where customers request. Therefore, we choose 15 drivers to satisfy the expected demand of 450 customers per day in the given time frame. The travel times are deterministic and derived from the Euclidean distance between location and destination. Travel times are rounded up to the next minute. We assume drivers to have a constant velocity of 30 km/h. The time to find a parking spot at a restaurant or customer is modeled by the normal distribution $\mathcal{N}(\mu = 2.5, \sigma = 0.5)$. Like travel times, parking times are rounded up to the next minute.

5.2 Parametrization of Methods

In the following, we provide an overview of the parametrizations of our methods and their impact.

5.2.1 Offline Method

We set the GBDT model to 1000 individual decision trees with a maximal depth of 5 and a maximum of 10 final nodes. Each final node contains a minimum of 400 training samples. The learning rate is set to 0.02. The hyperparameters were chosen manually and are not the result of hyperparameter optimization.

Training.

We train the GBDT until the validation loss does not improve over the course of 500 iterations. To train the GBDT, we simulate 1960 day (20 days on each of the 96 cores) with an average of 450 customers per day. We randomly split the data into 80% training samples and 20% validation samples. The learning curve for training the GBDT model is depicted in Fig. 6 (left). The x-axis shows the number of training iterations and the y-axis shows the MSE of the model on the validation samples. The dashed line represents the validation set and the solid line represents the training set. We observe a smooth convergence of the model and an negligible overfit of the training data. However, 1920 days of data might not be available in practice. To test if less data suffices, we plot the performance of the model in dependence of the number of training samples seen. For that purpose, we randomly sample subsets of size $\{1000, 2000, 3000, \ldots, 100000\}$. For each sample size, we train the model on

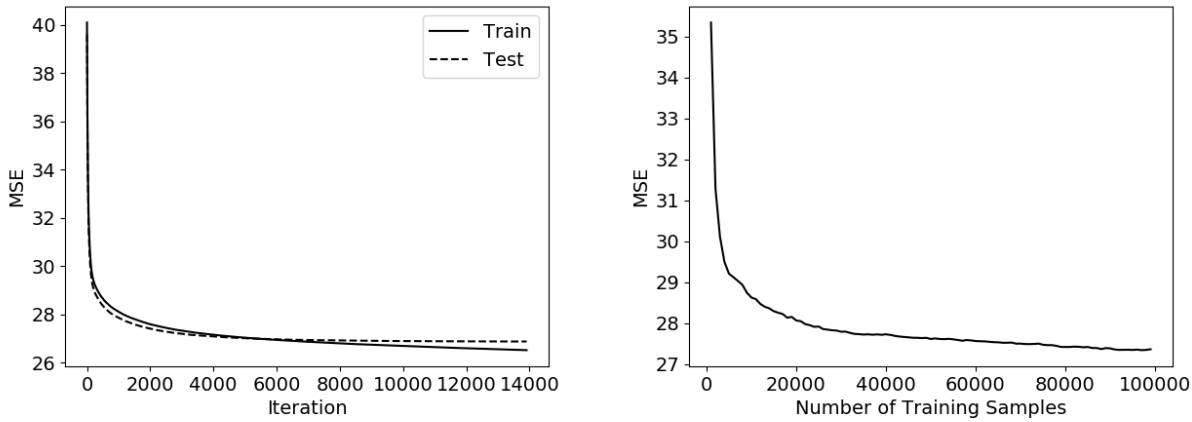

Figure 6 Learning Curve of GBDT Model (left) and MSE in Dependence of Number of Training Samples (right)

the corresponding subset to convergence. We validate the models on the same validation set. Fig. 6 (right) shows the result of our experiment. The x-axis denotes the size of the training set while the y-axis shows the MSE on the validation set. The figure suggests that it is sufficient to train the GBDT model with 100000 samples ($\approx$ 222 days) to achieve a performance comparable with our model that was trained on approximately 864000 samples (1920 days).

Features.

We use Shapley values to analyze the relative contribution of our handcrafted features to the model output. Shapley values origin from coalitional game theory. It is a method to fairly assign rewards to different parties depending on their contribution to the total payout. In our case, the parties are the features and the total payout is the model output. For a detailed explanation of Shapley values, the reader is referred to Molnar (2019). We evaluate the features with respect to their mean absolute Shapley value. Furthermore, we evaluate the feature importance by iteratively leaving out one feature and keeping all others in the training of our method. We illustrate the mean absolute Shapley value of each feature and the model performance without the feature in Fig. 7. The lower x-axis corresponds to the black bars and depicts the mean absolute Shapley value. The upper x-axis corresponds to the gray bars and depicts the MSE of the model without the feature. The feature names are

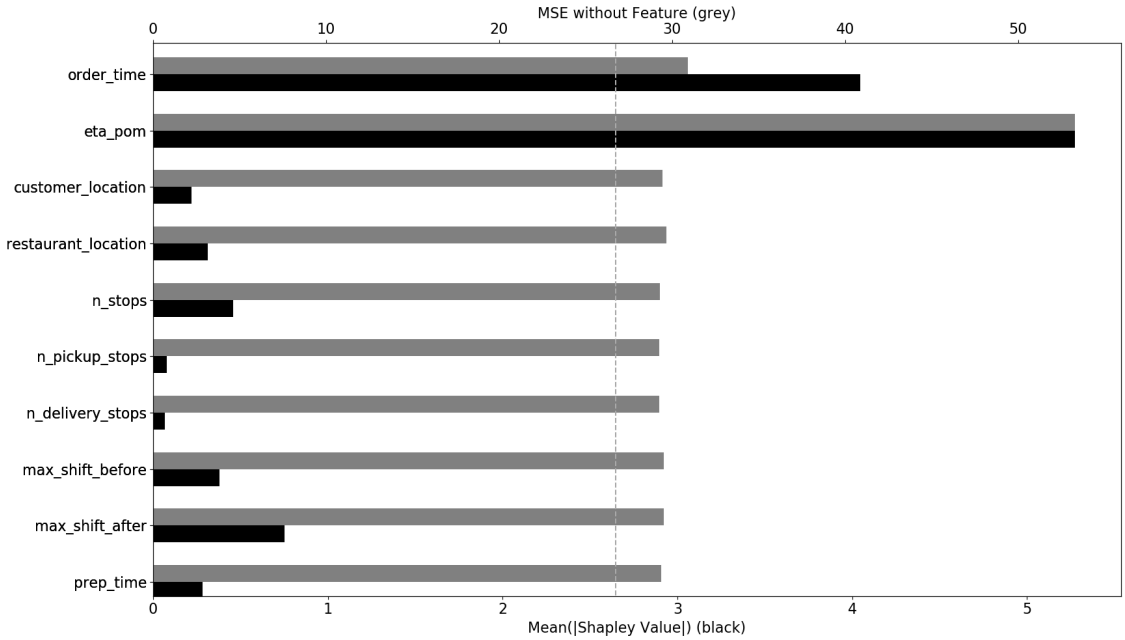

Figure 7 Mean Absolute Shapley Values and Model Performance without Feature

given by the y-axis. The vertical dashed line marks the MSE of the model using all features. All features are important as we can see by the gap between the model performance using all features (vertical dashed line) and the model performance when one feature is dropped (gray bars). However, we observe that temporal features (arrival time estimated by PoM and order time) have the highest impact on the model. In particular, the arrival time estimated by PoM is required to predict the residuals accurately. Without the feature, the MSE increases from 26.77 to 53.29. For all other features, we observe a less severe effect when dropping the feature.

5.2.2 Online-Offline Method

We train the DNN over 6 cycles of the cosine annealing learning rate strategy. We start with an initial cycle of 10 epochs and double the length of each subsequent cycle. In total, the network is trained for 630 epochs. During each cycle, we decay the learning rate from 0.001 to 0 by cosine annealing. A comparison of different learning rate strategies and the corresponding learning curves is entailed in Appendix *A*.2, Fig. 13. We use a batch size of

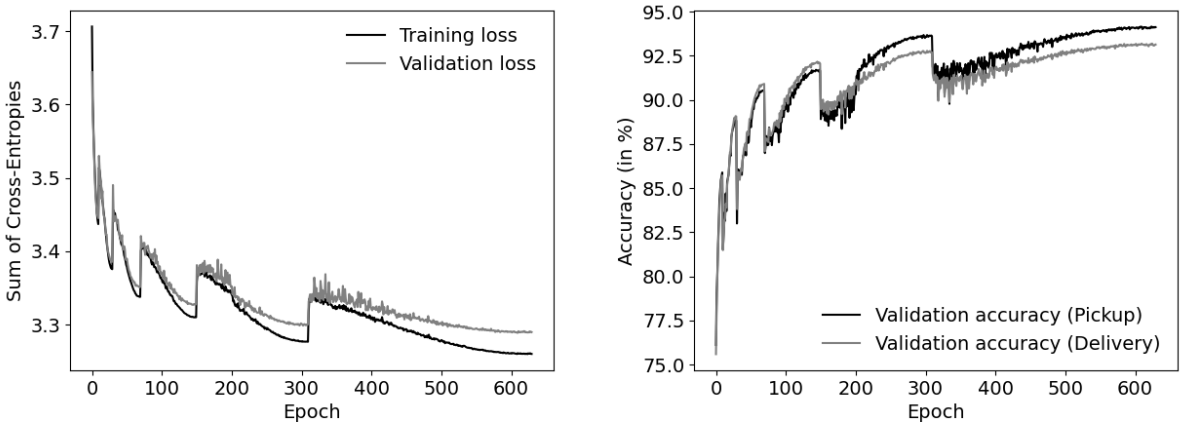

Figure 8 Training and Validation Cross-Entropy Loss (left) and Validation Accuracy of DNN for Predicting Insertion of Pickup and Delivery (right)

200 samples with each sample being a 65 dimensional vector. Again, we use 1920 training days and split them into 80% training and 20% validation data. The resulting cross-entropy loss for training and validation set is illustrated in Fig. 8 (left). The accuracy for predicting the insertion point for pickup and delivery on the validation set is depicted in Fig. 8 (right). The x-axis of both figures denotes the number of epochs trained. In the left figure, the black line shows the training loss and the grey line shows the validation loss. The y-axis of the left figure denotes the unweighted sum of cross-entropies for predicting pickup and delivery insertion. In the right figure, the black line corresponds to pickup insertion predictions and the gray line corresponds to delivery insertion predictions. On the y-axis, we show the mean accuracy on the validation set. The discontinuities in both graphs are the result of a sudden increase in learning rate caused by the cosine annealing learning rate strategy. We observe that the DNN provides very high accuracy. It predicts pickup insertions with an accuracy of 94% and delivery insertions with an accuracy of 93%.

5.3 Approximation Quality

In this section, we compare the quality of the different EAT policies. Each EAT policy leads to different arrival time estimations, different customer selections, and different future states. To ensure a fair comparison of all EAT policies, we do not communicate the arrival

time determined by each EAT policy but always use the arrival time derived by PoM in our experiments. This leads to the same EAT decision states for every policy (In Appendix A.3, we show that applying our offline method as EAT policy leads to rather minor differences in approximation quality even though the offline method is trained on the PoM data). We compare all policies on a validation instance of 50 days with an average of 450 customers per day.

In Section 4, we stated that our solution methods address the shortcomings of planning on means and full online simulations. Our offline method is designed to be as fast as PoM while tackling PoM's inherent myopia. Our online-offline method aims to reduce the computation time required for full online simulations while retaining their accuracy. Thus, we analyze the policies with respect to MSE and mean runtime per customer request.

The detailed results of our computational study are presented in Tab. 7 in Appendix A.4. An overview of the MSE and mean runtime of all methods is illustrated in Fig. 9. The x-axis states the policy employed. The left y-axis, with its corresponding black bars, denotes the MSE of the policy on the validation set and the right y-axis, with its corresponding gray bars, denotes the mean runtime in seconds to predict a single arrival time. We simulate $n = 32$ times in our online and online-offline method. We are limited to 32 simulations due to the high computational costs of the online method. We use the mean runtime of a single simulation and a single restaurant for our online-offline method and the online method. This is justified because multiple simulations might be performed in parallel (theoretically). Fig. 9 highlights the unfavorable accuracy-runtime trade-off of PoM and online simulations and the favorable trade-off of our proposed methods. PoM has low runtime but a high MSE and online simulations have a high runtime but a low MSE. In contrast, both of our proposed methods have low runtimes and low MSEs. Our offline method's computational time is comparable to planning on means. Both require less than a centisecond to predict arrival times. GBDT halves the MSE of planning on means and is, in contrast to planning on means, an unbiased estimator of the expected time of arrival (see the mean error in Tab. 7). Our online-offline method is an order of magnitude faster than exact online simulations. This is to be expected as we reduced the computational complexity from cubic to quadratic. Our policy also avoids extremely high runtimes of more than 10 seconds up to even more than

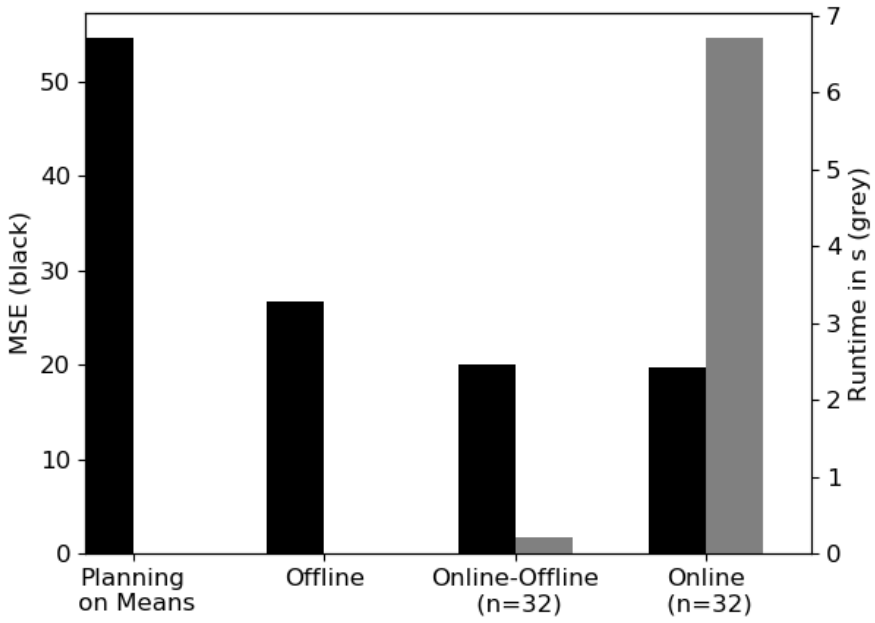

Figure 9 Comparison of MSE and Mean Runtime of EAT Policies

one minute per customer, restaurant, and simulation run observed for the online simulation (see Appendix A.2 for more detail). Our online-offline method's accuracy compares to the accuracy of full online simulations. However, the online-offline simulation comes with a small bias (see mean error in Tab. 7). The bias is likely caused by infeasible predictions of the DNN as we discussed in Section 4.3.

A more detailed insight on the quality of the methods is given by Fig. 10. In the figure, we compare the density estimation of the distribution of residuals of all EAT policies. The x-axis shows the prediction error in minutes. The y-axis shows the relative frequency of the error. We observe that PoM (solid line) has a second mode between 5 and 20 minutes. This indicates that PoM ignores bundling. The distribution of residuals of all other policies lack this second mode. This is a strong indicator that our proposed methods are accounting for uncertainty in bundling.

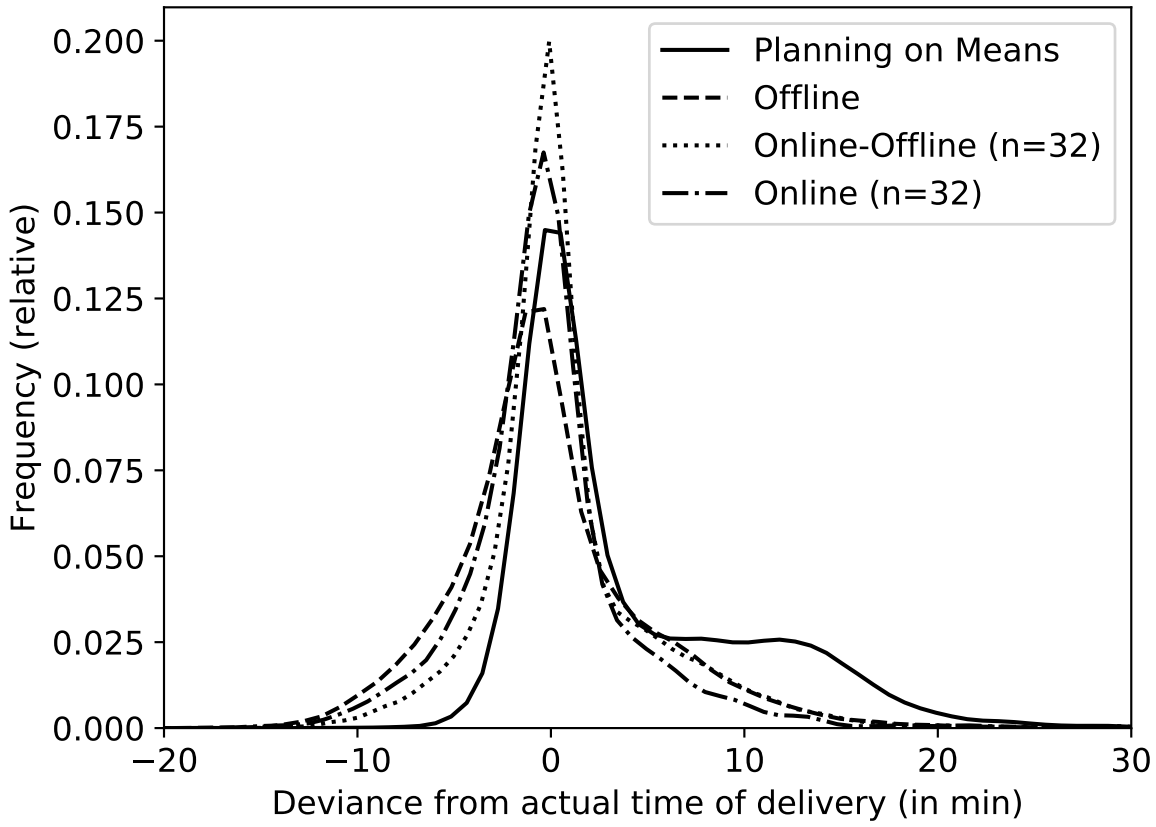

Figure 10 Comparison of Residuals of our Benchmark Policies over 50 Days

6 Analysis

In this section, we analyze the impact of employing our offline method on the RMDPEAT process. We do not consider our online-offline method as its higher runtime becomes noticeable when a great number of experiments have to be conducted. From a model point of view, we analyze the robustness of our offline method when employed on a setting dissimilar to the training setting. From a customer point of view, we analyze the effect of different choice models and customer locations. From the restaurant point of view, we analyze the impact of location, advertisement, and reliability on the restaurants' popularity and the platform's overall performance. Every experiment is conducted on a new instance of 1920 days with our offline method as EAT policy.

Table 2 Robustness of the Offline Method w.r.t. Changes in Restaurants, Fleet, and Demand

Instance Modifications	MSE (PoM)	MSE (Offline w/o Retraining)	MSE (Offline w/ Retraining)
Restaurants	53.02	26.59	26.45
Drivers	40.79	27.84	27.28
Demand	76.87	43.46	39.62
All	58.08	30.39	30.15

6.1 Ready For Growth.

Platforms always aim on growing their restaurant and customer base. We chose features for our GBDT model in Section 4.2 to allow for such changes in demand, fleet size, and number of restaurants (However, the small gap between offline method and online simulation indicates that the loss when choosing features independent of fleet size and number of restaurants is relatively small). The advantage of our feature selection is that we can apply our trained method even when business grows. In the following, we analyze how such changes impact our solution quality.

To this end, we probe the robustness of our offline method on four new test settings. In the first three settings, we increase either the fleet size to 20, available restaurants to 20, or demand to 585 expected customers per day (thus, by 30%); in the fourth setting we incorporate all three changes at once. We take our GBDT model from the standard instance described in Section 5.1 and employ it without retraining as EAT policy on the four test instances. For each instance, we use PoM and a GBDT model that was retrained on the new instance as benchmarks. The resulting MSE for each instance and each considered EAT policy is displayed in Tab. 2. As in the previous experiments, both offline methods always outperform PoM. There are no substantial differences between the MSE of our offline methods with and without retraining. The slight improvement in accuracy when retraining the method is most noticeably when the demand increases without an increase in drivers or restaurants. This might be explained by an increased likelihood of bundling. The results

further highlight the robustness of our proposed offline method. In particular, when business grows evenly the accuracy of our method remains constant. Thus, our method is well prepared to handle business growth.

6.2 The Role of the Customer

In this section, we analyze changes in customer behavior. So far, we assumed that customers choose from all restaurants but reject restaurants with an estimated time of arrival higher than 60 minutes. However, customers might be more selective in their restaurant choice or they might be more or less patient. We design two experiments to investigate the impact of different customer behavior on wait times of customers, on accuracy of arrival time estimations, on the chances of customers to order from their favorite restaurants, and on rate of rejection. Furthermore, we investigate how the location of a customer affects their service experience.

Patience Pays Off.

First, we analyze patience. To this end, we keep the instance setting but vary the delivery time threshold, i.e., the maximum time a customer is willing to wait for delivery. We consider five different settings from very impatient customer behavior (30 minutes) to very patient customer behavior (90 minutes) in steps of 15 minutes. Tab. 3 shows the obtained results. The first row states the time threshold (in minutes), i.e. the patience level of customers. The second row compares the MSE of the PoM and offline policy. The third row states the mean wait time of customers. The fourth row shows the percentage of customers that order at their favorite restaurant (the first in their list), second favorite restaurant, third favorite restaurant, or that did not order at all. Our offline method yields a lower MSE than PoM on all instances. We also observe that the prediction quality of our offline method is less affected by the different patience levels than PoM is. Customer wait time increases with the patience levels of customers but begins to plateau at a wait time of approximately 50 minutes. The initial increase in wait time is expected as higher patience levels also allow for longer trips and more orders. The convergence of wait times might be explained by the percent of customers that choose from their favorite restaurant: When customers are

Table 3 Customer Order Behavior For Different Time Constraints

Time Constraint		30	45	60	75	90
MSE (ETA)	PoM	28.1	49.0	57.7	60.7	61.7
	GBDT	17.1	28.6	31.1	29.5	28.6
Mean Wait Time		25.0	34.3	42.1	47.5	50.6
% of customers that ordered at	1st choice	36.3	53.2	70.2	85.2	94.9
	2nd choice	14.8	13.7	11.7	7.5	3.3
	3rd choice	8.8	7.6	5.5	2.9	0.9
	Rejected	13.3	6.0	1.9	0.4	0.0

willing to wait no more than 30 minutes, only 36.3% order at their favorite restaurant. In contrast, if customers are willing to wait up to one and a half hour 94.9% order from their favorite restaurant. The data suggests that the distribution of restaurant workloads converges to a uniform distribution when patience increases. Consequently, the mean wait time of customers also converges. We also observe a convergence of the rejection rate. The rejection rate decreases with increasing patience levels. 13.3% of customers willing to wait only 30 minutes do not order. When customers are willing to wait one hour for their meal, only 1.9% do not order. When the patience level further increases to one and a half hour, all customers order. Overall, we observe a strong impact of customer patience on served customers, average delivery times, and restaurant workload. Thus, knowing a customer's patience level might be very valuable for a platform.

You Can't Always Get What You Want.

Customers may be very obstinate in their restaurant choice; in an extreme scenario, a customer may only consider one restaurant to order from. Instead of modifying customers' patience level, we restrict the restaurants customers consider to a random subset in our next experiment. A subset of a given size is sampled from the set of restaurants individually for every customer. We test the impact of selective customers on 5 different settings with 1, 3, 5, 7, and 15 considered restaurants. The MSE, mean wait time, and rejection rate for different subset sizes are shown in Tab. 4. We see that the MSE and mean wait time increase when

Table 4 Customer Order Behavior Depending on Number of Restaurants a Customer Considers Ordering From

# Restaurants to choose from		1	3	5	7	15
MSE (ETA)	PoM	46.8	53.6	55.9	57.5	57.7
	GBDT	24.1	27.8	29.2	29.8	31.1
Mean Wait Time		38.4	40.6	41.2	41.5	42.1
% Rejected		13.2	6.3	4.2	3.2	1.9

the number of restaurants to choose from increases. Both observations might be connected to the increased rejection rate when only few options are given. As a result, fewer customers are served in total. This lowers the chance of bundling and allows for more accurate arrival time estimations. Fewer requests also lead to a lower utilization of the fleet, which allows for faster delivery. This might also explain why only 13.2% of customer reject to order when only one restaurant is considered compared to 29.8% of customers that did not order from their favorite restaurant in the last experiment (see Tab. 3). Nonetheless, a rejection rate of 13.8% due to very selective customers constitutes an enormous loss in revenue. Hence, platforms may think about countermeasures to keep picky customers happy.

Things Will Be Great When You're Downtown.

When customer are very selective in their restaurant choice, a customer that is not offered a "feasible" restaurant might have bad luck. However, even when customers are very open in their restaurant choice, the absence of feasible restaurants might be rooted in spatial discrimination. In the next experiment, we consider fairness aspects of restaurant meal delivery. Fig. 11 depicts the density estimation of ordering (left) and rejecting customers (right). The x- and y-axes denote the position in kilometers of the customers. Black squares represent restaurant locations. The contours illustrate different density levels that are specified by the number on the contour. The plots highlight that ordering and rejecting customers do not follow the same spatial distribution. While ordering customers are located prominently in downtown Iowa City, customers that rejected service are located disproportionately in the outskirts of the city. In particular the east of the city, where few restaurants are nearby, has

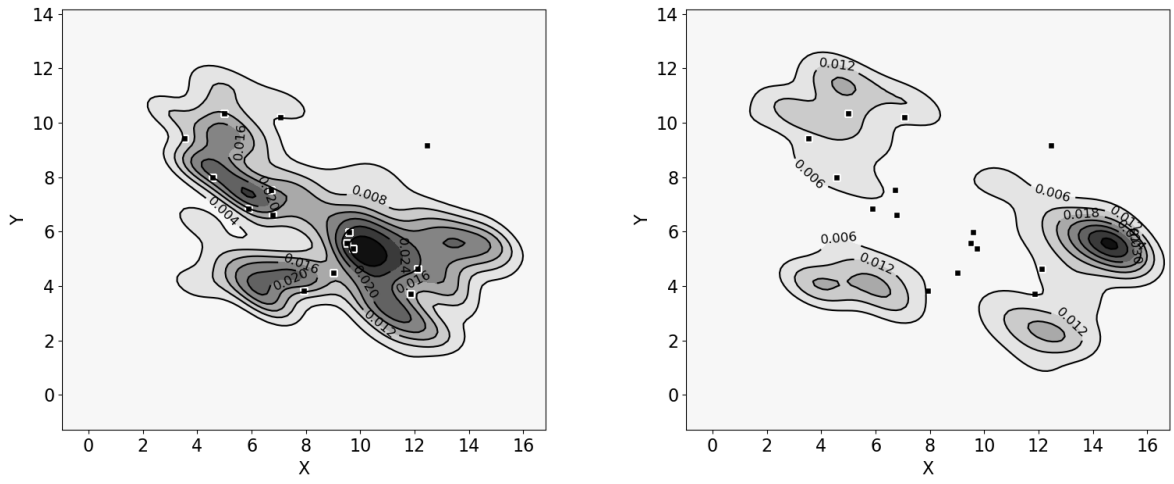

Figure 11 Kernel Density Estimation of Ordering Customers (left) and Rejecting Customers (right)

significantly more rejecting customers than other districts. We conclude that the location of a customer strongly determines the offered service level, and, as with many instant delivery applications, a systematic spatial discrimination can be observed (Howland 2016; Chen et al. 2020). Restaurant meal delivery platforms should therefore either limit their service area or focus on mechanisms to ensure service in the outskirts of the city, for example, by reserving vehicles for deliveries in these regions or by recruiting restaurants in areas with a high demand but low restaurant density. In the next section, we analyze the impact of location from the point of view of the restaurant.

6.3 The Role of the Restaurant

In this section, we analyze the effect of location, advertisement, and unreliability on a restaurant's popularity.

Location, Location, Location.

In the previous section, we observed a spatial bias towards downtown customers. We now show that this bias also holds for restaurants, even though the customer preferences are

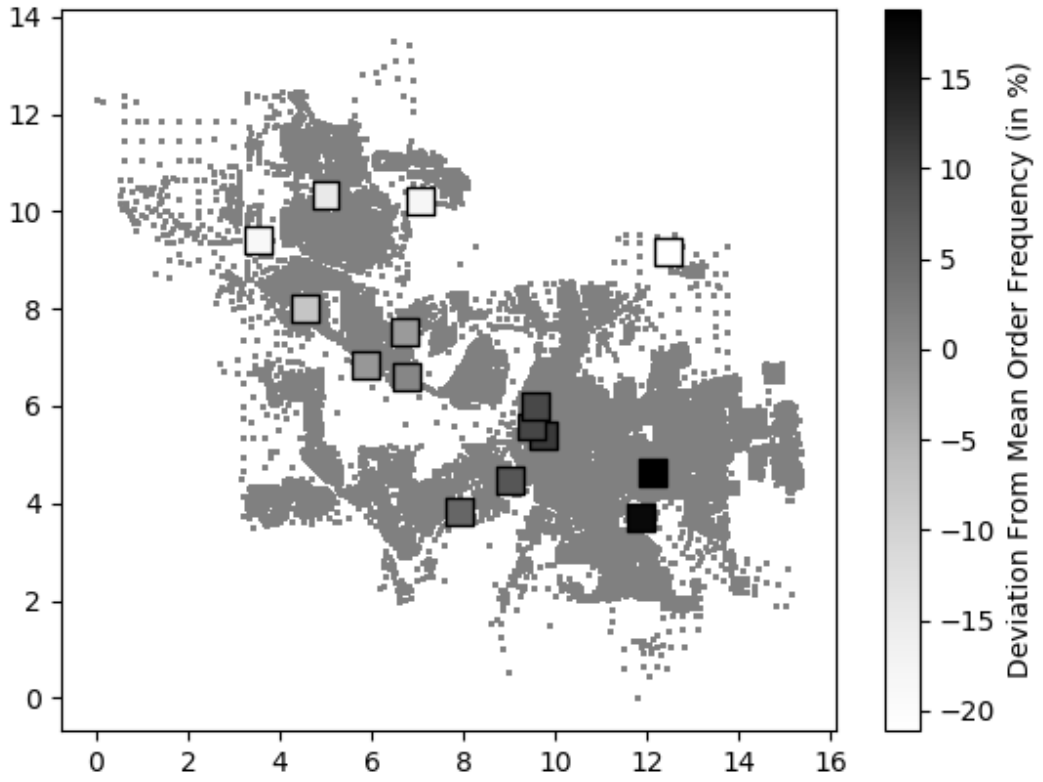

Figure 12 Spatial Distribution of Relative Deviation from Mean Restaurant Workload

homogeneous over the restaurants. To analyze the distribution of orders over restaurants, we calculate the relative deviation from mean restaurant workload of every restaurant. The results are illustrated in Fig. 12 following the structure of Fig. 5. The shade of the restaurant squares indicate the relative deviation of the restaurant's workload from the overall mean restaurant workload. We see that restaurants in the east-center of the city have workloads significantly higher than the mean workload. In contrast, restaurants in the northern outskirts of the city have a demand noticeably lower than the mean workload. Proximity of a restaurant to densely populated areas favors the restaurant due to low delivery times. Therefore, the observed inequalities can be explained by the high customer density in the city-center and low customer density in the northern parts of the city. The experiment stresses that restaurant workload is not uniformly distributed because of given spatial fac-

Table 5 Effect of Advertising One Restaurant

Restaurant Workload	Increase in Orders of Advertised Restaurant	Decrease in Overall Orders
Low	59.9% (+13.4)	0.1% (-0.5)
Average	51.9% (+15.0)	0.3% (-1.2)
High	38.9% (+13.2)	0.3% (-1.2)

tors. It also may guide platforms to recruit restaurants in high demand areas or even install their own kitchens.

The Impact of Advertisement.

In our main experiments, we assumed customers' restaurant preferences to be homogeneous. However, there are means to increase restaurant popularity, for example, by advertisement or low delivery fees (Glaeser et al. 2020). In the following, we analyze the impact of such increased popularity. To this end, we consider three settings. In each setting, we increase popularity for one restaurant; the restaurant with the lowest workload, a restaurant with an average workload, and the restaurant with the highest workload. In each setting, customers are twice as likely to choose the advertised restaurant compared to all other restaurants. The increase in orders for the respective restaurant and the overall relative and absolute decrease of orders per day are shown in Tab. 5. We observe that advertisement has the strongest relative effect on the restaurants with below-average workload. The relative effect decreases for the restaurant with average workload and is lowest for the restaurant with the highest workload. Surprisingly, all advertised restaurants had the same absolute daily increase in orders. We also observe an increased rejection rate when restaurants with a higher workload are advertised. An explanation could be their higher disparity in restaurant workloads. This effectively binds more vehicles to a single area and leaves other areas understaffed.

Unreliable Restaurants Jeopardize Themselves and the Platform.

In our main experiments, we assumed that all restaurants are equally reliable in their preparation times. In our final analysis, we test the effect of a restaurant that "misbehaves" by

Table 6 Effect of Increased Uncertainty for the Meal Preparation of One Restaurant

	Increase in MSE		Decrease in Orders	
Restaurant Workload	Restaurant	Overall	Restaurant	Overall
Low	66.2%	6.1%	7.4% (-1.6)	0.1% (-0.3)
Average	75.5%	9.0%	4.5% (-1.2)	0.2% (-1.1)
High	95.6%	16.7%	2.7% (-0.9)	0.2% (-1.2)

having high variance in their meal preparation times. In three nex experiments, we change the meal preparation process for the same three restaurants as in the previous experiment. One restaurant at a time, we assume preparation times to be given by $\mathcal{LN}(\mu = 7.41, \sigma = 1.6)$. The modified distribution has the same mean as the previous distribution of meal preparation times but a higher standard deviation. The relative increase in MSE and relative and absolute decrease in orders per day for the restaurant and all restaurants for all three instances is shown in Tab. 6. We observe a strong increase in MSE for estimated arrival times of high-variance restaurants. The higher the workload of the restaurant the higher the increase in MSE. This might be explained by the location of the restaurants and our Lemma 1. Restaurants with higher workloads are typically located in the center of the city. Therefore, the trips of drivers to the restaurants might be shorter on average. This increases the probability that wait times of drivers at the restaurant are affected by long food preparation times. In contrast, shorter food preparation times are still dominated by the time required to drive to the restaurant. Low-workload restaurants are typically located in the outskirts of the city. Trips to these restaurants might take longer on average and compensate for long meal preparation times. A decrease in reliability also comes with a decrease of orders for the restaurant and the platform. In particular, the already unpopular restaurant loses a significant portion of its customers. In contrast, the most popular restaurant loses only a small portion of its customers. We explain this observation as follows: If the workload of a unpopular restaurant in the outskirts of the city further decreases the chance of a vehicle being close to the restaurant also decreases. As a consequence, we can serve even fewer customer requests for this restaurant in a reasonable time. When we look at the decrease in orders for the platform, the trend is reversed: When a highly popular restaurant is unreliable,

the platform loses the most customers. In contrast, when an unpopular restaurant becomes unreliable, the platform loses only a few customers. This might be due to an increase in fleet flexibility when fewer vehicles drive to the low workload restaurant in the outskirts in the city. Therefore, we are able to better serve more densely populated areas in the city and are able to compensate for the customer loss at the low workload restaurant. However, the overall decrease in orders for the restaurant-meal-delivery platform is negligible. If a restaurant is unreliable customers just pick more reliable restaurants to order from. The increase in MSE, on the other hand, is likely to have longer-term effects on the restaurant's popularity and on the platform's performance.

7 Conclusion and Future Work

In this paper, we have presented solution methods to estimate arrival times for the restaurant meal delivery problem. Estimated arrival times influence customers in their restaurant choice, which, in turn, shapes future states of the restaurant meal delivery problem. We account for the interdependency of both problems by modeling them in a new unified Markov decision process. Estimating arrival times for restaurant meal deliveries is uniquely challenging due to the interlinked uncertainty in delivery process and meal preparation process, as well as the uncertainty in bundling of future demand.

To surmount these challenges, we have proposed an offline and an online-offline method. The offline method relies on gradient boosted decision trees that map a set of handcrafted features to estimated arrival times. The online-offline method exploits an offline approximation of the runtime-expensive routing policy that is learned supervised by a deep neural network to conduct online simulations of all processes, including future demand and routing decisions, in real time. We have tested our methods for restaurant meal delivery in Iowa City. Our results demonstrate that anticipation of the problem's dynamism and uncertainty is crucial to predict arrival times accurately. Our proposed methods efficiently incorporate the problem's dynamism and uncertainty, likely enabling them to perform close to a strong theoretical lower bound, while requiring minimal runtime.

Restaurant meal delivery platforms face the unique challenges of fast-paced last-mile

delivery with a constant need for optimization due to fierce competition in a rapidly growing market. This presents the restaurant meal delivery problem as a diverse and rich area of research in transportation and logistics. In the following, we discuss promising research avenues in this domain.

In our managerial analysis, we have shown that spatial factors, customer patience and flexibility, and advertisement strongly influence the dynamics of the problem. Building on the observed effect of restaurant advertisement, future research could extend the proposed model by considering dynamic pricing or advertising in order to deliberately increase the likelihood of logistically favorable future states. In this context, a closer look towards more detailed customer choice models may be very valuable. Since we have tested our method on a variety of different customer preference settings, we are confident that our method will likely perform well also in these cases. We also have shown the importance of restaurant and customer locations. Recently, restaurant meal delivery services started to open up kitchens that solely serve delivery requests. This initiates a new location planning problem, possibly even integrated in the presented problem, that has yet to be addressed in the literature. Furthermore, the heterogeneous service level raises fairness questions. Future work may analyze how fair service can be offered in an efficient way.

Methodologically, we have shown that routing decisions can be approximated with the help of a deep neural network to drastically accelerate online simulations of the delivery process. This online-offline approach is not limited to estimating arrival times, but constitutes a promising tool to accelerate all simulations that require routing decisions. Future work could exploit the online-offline framework to derive simulation-based anticipatory routing decisions for dynamic pickup and delivery problems, for example, in the field of passenger transportation, or urban same-day delivery.

Acknowledgement

This research was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - 413322447. We gratefully acknowledge their support. Furthermore, the authors acknowledge the North-German Supercomputing Alliance (HLRN) for providing HPC

resources that have contributed to the research results reported in this paper.

References

Bello, Irwan, Pham, Hieu, Le, Quoc V, Norouzi, Mohammad, and Bengio, Samy. 2016. "Neural combinatorial optimization with reinforcement learning." *arXiv preprint arXiv:1611.09940.*

Berbeglia, Gerardo, Cordeau, Jean-François, and Laporte, Gilbert. 2010. "Dynamic pickup and delivery problems." *European Journal of Operational Research* 202 (1): 8–15.

Bertsimas, Dimitris, Delarue, Arthur, Jaillet, Patrick, and Martin, Sébastien. 2019. "Travel time estimation in the age of big data." *Operations Research* 67 (2): 498–515.

Bissacco, Alessandro, Yang, Ming-Hsuan, and Soatto, Stefano. 2007. "Fast human pose estimation using appearance and motion via multi-dimensional boosting regression." In *2007 IEEE Conference on Computer Vision and Pattern Recognition,* 1–8. IEEE.

Chen, Mei, Liu, Xiaobo, Xia, Jingxin, and Chien, Steven. 2004. "A dynamic bus-arrival time prediction model based on APC data." *Computer-Aided Civil and Infrastructure Engineering* 19 (5): 364–376.

Chen, Xinwei, Wang, Tong, Thomas, Barrett W, and Ulmer, Marlin W. 2020. "Same-Day Delivery with Fairness." *arXiv preprint arXiv:2007.09541.*

Chu, Lianyu, Oh, S, and Recker, Will. 2005. "Adaptive Kalman filter based freeway travel time estimation." In *84th TRB Annual Meeting, Washington DC.*

Dai, Hongyan and Liu, Peng. 2019. "Workforce planning for O2O delivery systems with crowdsourced drivers." *Annals of Operations Research:* 1–27.

Davis, Mark M and Heineke, Janelle. 1998. "How disconfirmation, perception and actual waiting times impact customer satisfaction." *International Journal of Service Industry Management.*

Deudon, Michel, Cournut, Pierre, Lacoste, Alexandre, Adulyasak, Yossiri, and Rousseau, Louis-Martin. 2018. "Learning heuristics for the TSP by policy gradient." In *International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research,* 170–181. Springer.

Du, Mengnan, Liu, Ninghao, and Hu, Xia. 2019. "Techniques for interpretable machine learning." *Communications of the ACM* 63 (1): 68–77.

Florio, Alexandre M., Kinable, Joris, and Van Woensel, Tom. 2020. "Learning Time-Dependent Travel Speeds from Big Data." *Submitted.*

Friedman, Jerome H. 2001. "Greedy function approximation: a gradient boosting machine." *Annals of Statistics:* 1189–1232.

Glaeser, Chloe K., Oh, J., and Sue, X. 2020. "Online Food Ordering Platforms: Commission Rates and Delivery Fees." *Submitted.*

Hoogeboom, Maaike, Adulyasak, Yossiri, Dullaert, Wout, and Jaillet, Patrick. 2020. "The Robust Vehicle Routing Problem with Time Window Assignments." *To appear in Transportation Science.*

Howland, Daphne. 2016. *Amazon pledges same-day delivery to all urban neighborhoods after outcry.* `https://www.retaildive.com/news/amazon-pledges-same-day-delivery-to-all-urban-neighborhoods-after-outcry/418843/`. Online; accessed 27 July 2020.

Huang, He, Pouls, Martin, Meyer, Anne, and Pauly, Markus. 2020. "Travel Time Prediction using Tree-Based Ensembles." *arXiv preprint arXiv:2005.13818.*

Hutchinson, Rebecca A, Liu, Li-Ping, and Dietterich, Thomas G. 2011. "Incorporating boosted regression trees into ecological latent variable models." In *Twenty-fifth AAAI Conference on Artificial Intelligence.*

Ke, Guolin, Meng, Qi, Finley, Thomas, Wang, Taifeng, Chen, Wei, Ma, Weidong, Ye, Qiwei, and Liu, Tie-Yan. 2017. "Lightgbm: A highly efficient gradient boosting decision tree." In *Advances in Neural Information Processing Systems,* 3146–3154.

Kim, G. 2017. "Travel time estimation in vehicle routing problem." In *2017 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM),* 1004–1008. IEEE.

Kingma, Diederik P and Ba, Jimmy. 2014. "Adam: A method for stochastic optimization." *arXiv preprint arXiv:1412.6980.*

Lee, Wei-Hsun, Tseng, Shian-Shyong, and Tsai, Sheng-Han. 2009. "A knowledge based real-time travel time prediction system for urban network." *Expert systems with Applications* 36 (3): 4239–4247.

Lee, Ying. 2009. "Freeway travel time forecast using artifical neural networks with cluster method." In *2009 12th International Conference on Information Fusion,* 1331–1338. IEEE.

Li, Xiangyong, Tian, Peng, and Leung, Stephen CH. 2010. "Vehicle routing problems with time windows and stochastic travel and service times: Models and algorithm." *International Journal of Production Economics* 125 (1): 137–145.

Lingitz, Lukas, Gallina, Viola, Ansari, Fazel, Gyulai, Dávid, Pfeiffer, András, Sihn, Wilfried, and Monostori, László. 2018. "Lead time prediction using machine learning algorithms: A case study by a semiconductor manufacturer." *Procedia CIRP* 72:1051–1056.

Liu, Sheng, He, Long, and Shen, Zuo-Jun Max. 2018. "On-time last mile delivery: Order assignment with travel time predictors." *Forthcoming in Management Science.*

Loshchilov, Ilya and Hutter, Frank. 2016. "SGDR: Stochastic gradient descent with warm restarts." *arXiv preprint arXiv:1608.03983.*

Mańdziuk, Jacek. 2018. "New shades of the vehicle routing problem: emerging problem formulations and computational intelligence solution methods." *IEEE Transactions on Emerging Topics in Computational Intelligence* 3 (3): 230–244.

Miki, Shoma, Yamamoto, Daisuke, and Ebara, Hiroyuki. 2018. "Applying deep learning and reinforcement learning to traveling salesman problem." In *2018 International Conference on Computing, Electronics & Communications Engineering (iCCECE),* 65–70. IEEE.

Molnar, Christoph. 2019. *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable.* https://christophm.github.io/interpretable-ml-book/.

Pittman, Simon J and Brown, Kerry A. 2011. "Multi-scale approach for predicting fish species distributions across coral reef seascapes." *PLOS ONE* 6 (5).

Polato, Mirko, Sperduti, Alessandro, Burattin, Andrea, and Leoni, Massimiliano de. 2014. "Data-aware remaining time prediction of business process instances." In *2014 International Joint Conference on Neural Networks (IJCNN),* 816–823. IEEE.

Poschmann, Peter, Weinke, Manuel, Balster, Andreas, Straube, Frank, Friedrich, Hanno, and Ludwig, André. 2019. "Realization of ETA Predictions for Intermodal Logistics Networks Using Artificial Intelligence." In *Interdisciplinary Conference on Production, Logistics and Traffic,* 155–176. Springer.

Powell, Warren B. 2011. *Approximate Dynamic Programming: Solving the curses of dimensionality.* Vol. 703. John Wiley & Sons.

Prokhorchuk, Anatolii, Dauwels, Justin, and Jaillet, Patrick. 2019. "Estimating Travel Time Distributions by Bayesian Network Inference." *IEEE Transactions on Intelligent Transportation Systems* 21 (5): 1867–1876.

Reimann, Martin, Lünemann, Ulrich F, and Chase, Richard B. 2008. "Uncertainty avoidance as a moderator of the relationship between perceived service quality and customer satisfaction." *Journal of Service Research* 11 (1): 63–73.

Reyes, Damian, Erera, Alan, Savelsbergh, Martin, Sahasrabudhe, Sagar, and O'Neil, Ryan. 2018. "The meal delivery routing problem." *Optimization Online.*

Smith, Leslie N. 2017. "Cyclical learning rates for training neural networks." In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV),* 464–472. IEEE.

Statista Survey. 2020. *eServices Report 2020 - Online Food Delivery.* https://www.statista.com/study/40457/food-delivery/.

Thompson, David A, Yarnold, Paul R, Williams, Diana R, and Adams, Stephen L. 1996. "Effects of actual waiting time, perceived waiting time, information delivery, and expressive quality on patient satisfaction in the emergency department." *Annals of Emergency Medicine* 28 (6): 657–665.

Ulmer, Marlin. 2020. "Dynamic pricing and routing for same-day delivery." *Transportation Science* 54 (4): 1016–1033.

Ulmer, Marlin W, Goodson, Justin C, Mattfeld, Dirk C, and Hennig, Marco. 2019. "Offline–online approximate dynamic programming for dynamic vehicle routing with stochastic requests." *Transportation Science* 53 (1): 185–202.

Ulmer, Marlin and Savelsbergh, Martin. 2020. "Workforce Scheduling in the Era of Crowdsourced Delivery." *Transportation Science* 54 (4): 1113–1133.

Ulmer, Marlin and Thomas, Barrett W. 2019. "Enough Waiting for the Cable Guy—Estimating Arrival Times for Service Vehicle Routing." *Transportation Science* 53 (3): 897–916.

Ulmer, Marlin, Thomas, Barrett W, Campbell, Ann Melissa, and Woyak, Nicholas. 2020. "The restaurant meal delivery problem: Dynamic pick-up and delivery with deadlines and random ready times." *To appear in Transportation Science.*

Van Lint, JWC, Hoogendoorn, SP, and Zuylen, Henk J van. 2005. "Accurate freeway travel time prediction with state-space neural networks under missing data." *Transportation Research Part C: Emerging Technologies* 13 (5-6): 347–369.

Vareias, Anastasios D, Repoussis, Panagiotis P, and Tarantilis, Christos D. 2019. "Assessing customer service reliability in route planning with self-imposed time windows and stochastic travel times." *Transportation Science* 53 (1): 256–281.

Wang, Dehua, Zhang, Yang, and Zhao, Yi. 2017. "LightGBM: an effective miRNA classification method in breast cancer patients." In *Proceedings of the 2017 International Conference on Computational Biology and Bioinformatics,* 7–11.

Wu, Fan and Wu, Lixia. 2019. "DeepETA: A Spatial-Temporal Sequential Neural Network Model for Estimating Time of Arrival in Package Delivery System." In *Proceedings of the AAAI Conference on Artificial Intelligence,* 33:774–781.

Yildiz, Baris and Savelsbergh, Martin. 2019. "Provably high-quality solutions for the meal delivery routing problem." *Transportation Science* 53 (5): 1372–1388.

Zhang, Junlong, Lam, William HK, and Chen, Bi Yu. 2013. "A stochastic vehicle routing problem with travel time uncertainty: trade-off between cost and customer service." *Networks and Spatial Economics* 13 (4): 471–496.

Zhu, Lin, Yu, Wei, Zhou, Kairong, Wang, Xing, Feng, Wenxing, Wang, Pengyu, Chen, Ning, and Lee, Pei. 2020. "Order Fulfillment Cycle Time Estimation for On-Demand Food Delivery." In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining,* 2571–2580.

A Appendix

In Appendix A.1, we analyze the role of bundling in RMDPEAT. In Appendix A.2, we compare different learning rate strategies to train our DNN (see Section 4.3 and Section 5.2). In Appendix A.3, we employ our offline method as EAT policy. In Appendix A.4, we provide a Table with detailed results on the approximation quality of our methods and a graphical comparison of the runtime distribution of online and online-offline simulations (see Section 5.3).

A.1 No Bundling

In our work, we claimed that bundling is a major source of delay. We further claimed that bundling is necessary to assert efficiency of our fleet. In the following, we conduct two experiments on a standard instance of 1920 days. In the first experiment, we use PoM as EAT policy and in the second experiment, we use our offline method as EAT policy. We use a first-in-first-out routing policy to avoid bundling entirely. We observe an MSE of 5.36 when using PoM and an MSE of 4.57 when using our offline method. Both MSEs are significantly lower than when we allow bundling (see Section 5.3). This indicates that bundling is a major cause of delay that is hard to predict. However, when disallowing bundling, we see a rejection rate of 13.9% for PoM and a rejection rate of 14.0% for our offline method. This is a considerable loss in customers compared to a rejection rate of 1.9% when allowing bundling. We conclude that bundling makes it difficult to predict arrival times but is necessary to serve customers efficiently.

A.2 Comparison of Learning Rate Strategies

When training our DNN, we considered three different learning rate strategies. We test a constant learning rate, a cyclic super-convergence learning rate strategy as given by Smith (2017), and a cosine annealing learning rate strategy as discussed in Section 4.3. Fig. 13 shows the result of our experiments. The left figure describes the learning rate (y-axis) for each batch iteration (x-axis). The right figure describes the cross-entropy loss (y-axis) over 150 epochs (x-axis) for our DNN for each learning rate strategy. We observe that

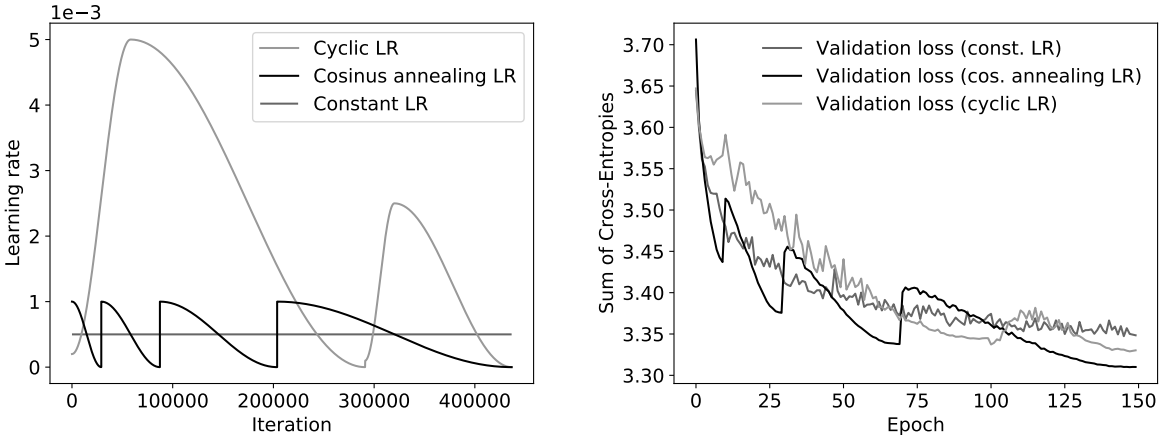

Figure 13 Learning Rate Strategies (left) and Cross-Entropy Loss for Different Learning Rate Strategies (right)

the cosine annealing learning rate strategy and the cyclic super-convergence learning rate strategy outperform the constant learning rate strategy. The DNN using a cosine annealing learning rate strategy has a lower cross-entropy loss than the DNN using a cyclic super-convergence learning rate strategy when trained for 150 epochs. For this reason, we decided to train our DNN with the cosine annealing learning rate strategy.

A.3 Robustness of our Offline Method

In the previous section, we only communicated arrival times derived by PoM. Our offline method did not interfere with the RMDPEAT process. Communicating different arrival times changes the data and therefore might reduce accuracy of our policy. In the following, we employ the offline method as EAT policy on our standard instance. We observe a significant increase in prediction quality, a negligible decrease in overall orders, and a noticeable decrease in mean wait time when employing the offline method:

1. The MSE decreases from 57.74 for PoM to 31.05 for our offline method. The loss in accuracy when applying the offline method as EAT policy in contrast to just predicting with it without communicating the arrival times to customer is negligible. The offline method still significantly outperforms PoM in terms of accuracy.

2. We observe a decrease of 0.5% in orders. We proved that PoM underestimates arrival

times in Lemma 1 and confirmed the Lemma and the negligible bias of our offline method empirically in Section 5.3. Therefore, the cause of the slight decrease in orders might be higher arrival time estimations communicated by our offline method compared to the arrival times communicated by PoM.

3. When employing the offline method the average wait time of customers decreases from 45.12 minutes to 42.08 minutes. This indicates that a higher accuracy in arrival time estimation might lead to lower average wait times for customers. An explanation could be that an anticipatory EAT policy induces a logistically more favorable distribution of orders.

A.4 Approximation Quality

We show the detailed experimental results on the test instances of Section 5 in Tab. 7. We performed up to 128 online-offline simulations, but only up to 32 online simulations. A higher number of online simulations were not possible due to their heavy computational burden. The runtime of online and online-offline simulations are illustrated in Fig. 14. The x-axis denotes the runtime required in seconds of each simulation. The y-axis denotes the relative frequency of each runtime. The distribution is estimated by a Gaussian kernel density estimator. We observe that the mean and variance of our online-offline simulation's runtime distribution are by an order of magnitude smaller than the mean and variance of the online simulation's runtime distribution.

Table 7 Quality of Arrival Time Estimations

Method		MSE	MAE	ME	Time (in s/cust.)
Offline Methods					
Planning on means		57.46	4.83	4.16	0.00
GBDT		26.67	3.70	-0.07	0.00
Combination of Offline and Online Methods					
	n=				
Approx. simul.	1	32.42	3.57	0.78	0.21
	2	25.89	3.24	0.79	
	4	22.79	3.05	0.78	
	8	21.31	2.94	0.78	
	16	20.47	2.88	0.80	
	32	20.05	2.84	0.79	
	64	19.83	2.83	0.79	
	128	19.77	2.83	0.79	
Online Methods					
	n=				
Exact simul.	1	35.70	3.67	-0.23	6.71
	2	28.76	3.32	-0.24	
	4	24.18	3.09	-0.24	
	8	21.42	2.93	-0.24	
	16	20.42	2.87	-0.23	
	32	19.81	2.81	-0.18	

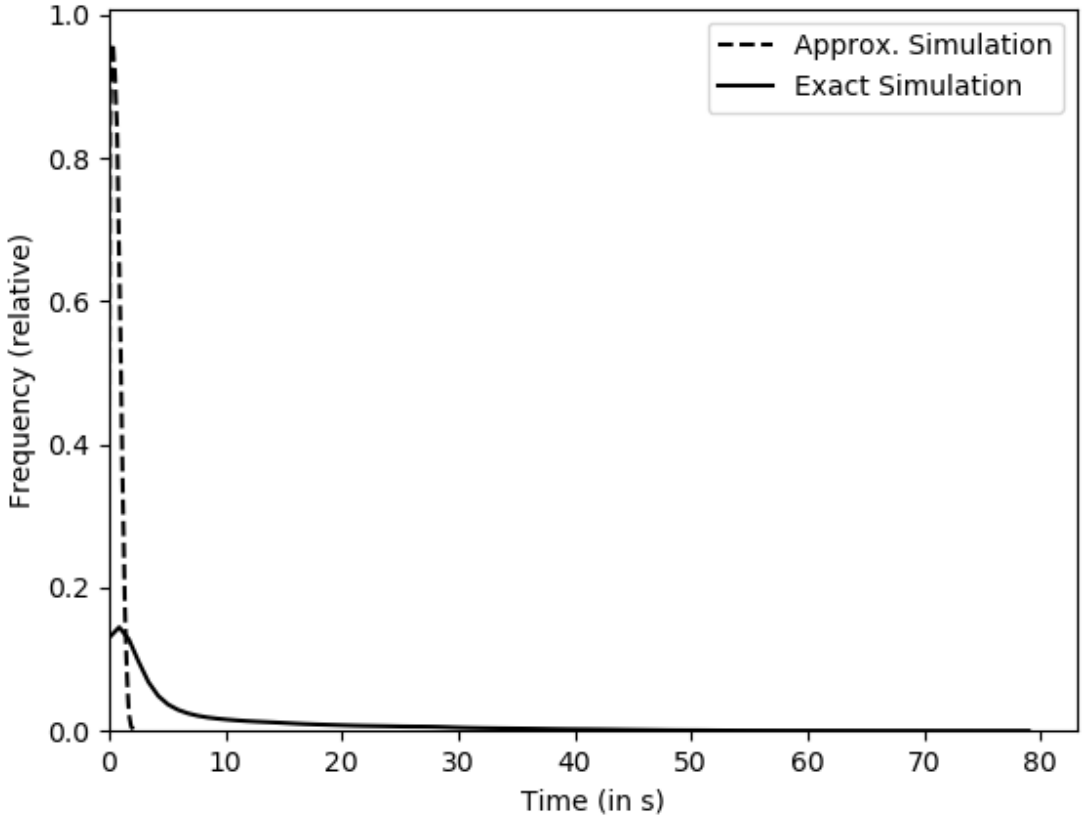

Figure 14 Comparison of Mean Runtime of Online and Online-Offline Simulations (n=1)