



Transportation Science

Publication details, including instructions for authors and subscription information:
<http://pubsonline.informs.org>

The Restaurant Meal Delivery Problem: Dynamic Pickup and Delivery with Deadlines and Random Ready Times

Marlin W. Ulmer, Barrett W. Thomas, Ann Melissa Campbell, Nicholas Woyak

To cite this article:

Marlin W. Ulmer, Barrett W. Thomas, Ann Melissa Campbell, Nicholas Woyak (2020) The Restaurant Meal Delivery Problem: Dynamic Pickup and Delivery with Deadlines and Random Ready Times. Transportation Science

Published online in Articles in Advance 27 Aug 2020

. <https://doi.org/10.1287/trsc.2020.1000>

Full terms and conditions of use: <https://pubsonline.informs.org/Publications/Librarians-Portal/PubsOnLine-Terms-and-Conditions>

This article may be used only for the purposes of research, teaching, and/or private study. Commercial use or systematic downloading (by robots or other automatic processes) is prohibited without explicit Publisher approval, unless otherwise noted. For more information, contact permissions@informs.org.

The Publisher does not warrant or guarantee the article's accuracy, completeness, merchantability, fitness for a particular purpose, or non-infringement. Descriptions of, or references to, products or publications, or inclusion of an advertisement in this article, neither constitutes nor implies a guarantee, endorsement, or support of claims made of that product, publication, or service.

Copyright © 2020, INFORMS

Please scroll down for article—it is on subsequent pages






With 12,500 members from nearly 90 countries, INFORMS is the largest international association of operations research (O.R.) and analytics professionals and students. INFORMS provides unique networking and learning opportunities for individual professionals, and organizations of all types and sizes, to better understand and use O.R. and analytics tools and methods to transform strategic visions and achieve better outcomes.

For more information on INFORMS, its publications, membership, or meetings visit <http://www.informs.org>

The Restaurant Meal Delivery Problem: Dynamic Pickup and Delivery with Deadlines and Random Ready Times

Marlin W. Ulmer,^a Barrett W. Thomas,^b Ann Melissa Campbell,^b Nicholas Woyak^b

^a Carl-Friedrich-Gauß-Fakultät, Technische Universität Braunschweig, Braunschweig, Germany 38106; ^b Tippie College of Business, University of Iowa, Iowa City, Iowa 52242

Contact: m.ulmer@tu-braunschweig.de,  <https://orcid.org/0000-0003-2499-6570> (MWU);  <https://orcid.org/0000-0002-6080-6191> (BWT); barrett-thomas@uiowa.edu,  <https://orcid.org/0000-0002-9927-687X> (AMC); nicholasrwoyak@gmail.com (NW)

Received: October 12, 2017

Revised: December 28, 2018; November 8, 2019

Accepted: April 8, 2020

Published Online in Articles in Advance:
August 27, 2020

<https://doi.org/10.1287/trsc.2020.1000>

Copyright: © 2020 INFORMS

Abstract. We consider a stochastic dynamic pickup and delivery problem in which a fleet of drivers delivers food from a set of restaurants to ordering customers. The objective is to dynamically control a fleet of drivers in a way that avoids delays with respect to customers' deadlines. There are two sources of uncertainty in the problem. First, the customers are unknown until they place an order. Second, the time at which the food is ready at the restaurant is unknown. To address these challenges, we present an anticipatory customer assignment (ACA) policy. To account for the stochasticity in the problem, ACA postpones the assignment decisions for selected customers, allowing more flexibility in assignments. In addition, ACA introduces a time buffer to reduce making decisions that are likely to result in delays. We also consider bundling, which is the practice of assigning multiple orders at a time to a driver. Based on real-world data, we show how ACA is able to improve service significantly for all stakeholders compared with current practice.

Supplemental Material: The online appendix is available at <https://doi.org/10.1287/trsc.2020.1000>.

Keywords: restaurant meal delivery • dynamic vehicle routing • stochastic requests • stochastic ready times

1. Introduction

People like to eat at home but do not always like to prepare their meals themselves. In a recent survey, 68% of respondents reported ordering takeout at least once a month with 33% ordering at least once a week (Statista Survey 2016). Seeking to satisfy this demand, recent years have seen a surge in companies that deliver restaurant meals to customers. These companies include startups such as DoorDash, Grubhub, SkipTheDishes, and UberEats. These companies offer customers the chance to place an order via a mobile application or internet website and choose from the full menu of dozens of restaurants. The delivery fee is typically between \$4 and \$7, and the food usually arrives between 30 and 40 minutes after the customer's request (Macmillan 2016). The global market for food delivery is expected to grow 20% a year to \$365 billion by 2030 from just \$35 billion today (Cheng 2018).

To be successful, restaurant meal delivery companies must satisfy several key stakeholders that are involved in the meal delivery transaction. Customers want reliable and fast service. Cooperating restaurants want their product served fresh to satisfy customers and ultimately to grow their customer base. Finally, drivers want to serve enough orders to make a decent wage.

Satisfying these stakeholders creates significant operational challenges for the delivery companies. Cooperating restaurants often pay a fee to the meal

delivery company for each order. This fee serves as compensation for offering the delivery service and reduces the delivery fee paid by customers. On the surface, this is beneficial to the restaurants when they see an increase in sales that justifies the fees. However, late deliveries or meals that are no longer fresh can make customers dissatisfied and affect the delivery company's partnerships. For example, in a recent interview, Kurt Kane, Wendy's chief concept and marketing officer, said that DoorDash had become the company's lead delivery partner because "the food almost always arrived hot" (Zaleski 2018). Further, when orders arrive late, not only are customers less likely to use the delivery service again, but they are also less likely to patronize the restaurant in any form (Maze 2016). Thus, late deliveries can cause cooperating restaurants to receive a bad reputation and eventually pull their partnership.

An easy way to prevent some undesirable late deliveries is to simply have more drivers available to perform deliveries. However, because drivers are paid a fee per delivery, they must make enough deliveries in an hour for it to be financially worthwhile for them to work (Isaac 2016). As drivers are added, the number of orders per driver goes down, and each driver earns less. When the number of deliveries per hour is too small, companies face increased costs for two reasons. Either they need to recruit and train new

drivers as existing drivers leave the service, or they must pay more to existing drivers to compensate current drivers for the loss in volume.

With these operational challenges in mind, we introduce the restaurant meal delivery problem (RMDP). The RMDP is characterized by a fleet of delivery vehicles that serve dynamic customer requests over the course of a day. The probability distributions on the time and location of customer requests are known. Customers make their requests (e.g., an order for food), choosing from a number of known restaurants throughout their area. The probability distribution on customer restaurant selection is also known. Once a request is placed, the order is relayed to the delivery company, which assigns it to a driver who picks up and delivers the order. The assignments of orders to drivers need not be immediate, and it is possible to bundle orders. We use the term “bundling” to refer to the practice of assigning multiple orders at a time to a driver. Having multiple assigned orders at once, a driver can choose to visit restaurants nearby to one another before making the food deliveries to the customers.

Before delivery of an order, the driver must pick up the order from the associated restaurant. However, the time to prepare a customer’s food at each restaurant is random. Although the delivery company knows a distribution on that time, the delivery company does not know exactly when the order will be ready. Thus, the driver may need to wait for the order’s completion when arriving to a restaurant.

Once the order has been retrieved, the driver delivers it to the customer, who expects the order to be delivered by a certain deadline. The objective is to minimize the expected delays, the sum of positive differences between delivery times and deadlines of orders over the service day. For example, consider a request placed at 5:00 p.m. with a deadline of 5:40 p.m. An order arriving at any time 5:40 p.m. or earlier would contribute nothing to the objective, and an order arriving at 5:45 p.m. would incur a penalty of five.

Minimizing delays requires effective deployment of drivers. Ideally, drivers arrive to the restaurant just as the food is ready and immediately depart for the customer. In practice, however, the numerous random elements of the problem make such planning a challenge.

Our conversations with a service provider reveal that many currently assign the new order to the driver they think is able to deliver the order the fastest. This policy is myopic and neither accounts for future orders nor random ready times. Further, sending the closest driver may limit the opportunities to assign orders to drivers that are already en route to a nearby restaurant.

To account for the sources of randomness and to enable bundling operations, we develop an anticipatory

customer assignment (ACA) policy. This policy combines a postponement strategy with a temporal buffer. The postponement of orders allows the delivery company to wait for additional information with respect to ready times and new orders before assigning a driver to an order. The time buffer allows the delivery company to hedge against potential delays caused by ready times that are later than expected and the insertion of new requests into the drivers’ routes. The buffer also serves as an instrument to control fleet utilization by balancing the trade-off between immediate service and the flexibility to serve future orders without delay.

We analyze our method using instances based on a restaurant meal delivery service located in Iowa City. Iowa City has features in common with many medium-sized cities where meal delivery occurs and mimics a single delivery zone within larger cities. Comparisons with the current practice and other benchmarks show that ACA results in significant advantages with respect to all stakeholders’ objectives. From an algorithmic perspective, we show the following:

- The proposed policy outperforms current practice as well as other intuitive policies.
- The buffer and postponement policies work together to overcome the uncertainty in the problem.
- The proposed approach can effectively counter heterogeneous ready-time distributions.
- The proposed approach is capable of scaling to handle larger-sized implementations.

Managerially, we show the following:

- For customers, our policy significantly reduces the average and maximum delays.
- For restaurants, our policy also avoids delivery of exceptionally unfresh food, whereas such unfresh deliveries are regularly observed by the current policy used in practice. The improved freshness of the delivered food leads to potentially more orders in the future as customers are likely to be happier.
- For drivers, our policy maintains the fairness among drivers in that, over time, all drivers deliver almost the same number of orders. We further show that, with our policy, the same fleet size is able to achieve less delay than current practice even as the number of orders per driver increases.
- For the meal delivery company, the policy supports growth by allowing the delivery company to add more trips with the same number of drivers and, thus, increase the revenue per driver.

We also contribute a set of data sets to promote further study of the RMDP. They are available at http://ir.uiowa.edu/tippie_pubs/69.

The outline of the paper is as follows. In Section 2, we present the related literature. We define the RMDP in Section 3 and the solution method in Section 4. Section 5 details the instances and implementation

that we use to perform our computational study. The section also introduces the benchmarks we use to compare the proposed ACA policy. Section 6 analyzes the computational performance of the proposed algorithm, and Section 7 details the impact of ACA on the stakeholders. The paper concludes with a summary and an outlook for future research opportunities.

2. Literature Review

In this section, we present a review of literature related to the RMDP. The RMDP is most similar to the dynamic pickup and delivery problem (DPDP) and the same-day delivery problem (SDDP). We spend the majority of our review on these problems. We also discuss literature related to customer service-oriented objectives in routing problems as well as papers that consider ready times in routing.

Table 1 summarizes the DPDP and SDDP literature related to the RMDP. The first column sorts papers by problem type. The second column provides the author and year of the paper. The remaining columns indicate particular features of the problems addressed in the papers. The RMDP is a dynamic problem with deadlines at customers, stochastic ready times, a customer service-focused objective function, and a large number of vehicles serving a large number of orders. Throughout this literature review, we consider only papers in which there are dynamic customer requests and, thus, do not identify this attribute in the table. The column titled “Deadlines” indicates whether the

paper in question addresses customer deadlines, which may be in the form of time windows. The column “Timing” indicates whether the timing of a visit to a customer is impacted by problem elements other than the dynamic requests of customers. Examples of such problem elements are random ready times and random travel or service times. The column “Objective” indicates whether the objective of the paper focuses on customer service. The most likely alternative is an objective focused on the delivery company’s costs or its revenue. The column “Postponement” identifies whether the paper allows the assignment of requests to a vehicle to be postponed to allow for potentially better bundling of orders. The column “Scalability” refers to whether the method in the associated paper can scale to handle a large number of customers and vehicles. Finally, the column “Anticipation” identifies whether the proposed solution method incorporates anticipation. By anticipation, we are referencing approximate dynamic programming techniques, such as value function approximation (including lookahead), policy function approximation, or cost function approximation. Such approximations require learning, such as approximate value iteration, reinforcement learning, or direct policy search. Powell (2011) provides an overview.

2.1 Dynamic Pickup and Delivery

The DPDP is the problem that arises when goods must be transported from unique pickup locations to

Table 1. Literature Classification

	Paper	Deadlines	Timing	Objective	Postponement	Scalability	Anticipation
DPDP	Mitrović-Minić and Laporte (2004)	✓				✓	
	Mitrović-Minić, Krishnamurti, and Laporte (2004)	✓				✓	
	Cortés, Núñez, and Sáez (2008)	✓	✓	✓			✓
	Sáez, Cortés, and Núñez (2008)	✓		✓			✓
	Cortés et al. (2009)	✓		✓			✓
	Fagerholt, Foss and Horgen (2009)	✓				✓	
	Ghiani et al. (2009)	✓		✓			✓
	Schilde, Doerner, and Hartl (2011)	✓	✓	✓			✓
	Hyytiä, Penttinen, and Sulonen (2012)	✓		✓		✓	✓
	Núñez et al. (2014)	✓		✓			✓
	Muñoz Carpintero et al. (2015)	✓		✓			✓
	Schilde, Doerner, and Hartl (2014)	✓	✓	✓			✓
	Sayarshad and Chow (2015)	✓		✓		✓	✓
	Vonolfen and Affenzeller (2016)	✓				✓	✓
	Ghiani, Manni, and Romano (2018)	✓		✓		✓	✓
	Reyes, Erera, and Savelbergh (2018)	✓	✓	✓	✓	✓	
SDDP	Azi, Gendreau, and Potvin (2012)	✓			✓		✓
	Ulmer, Thomas, and Mattfeld (2019)						✓
	Voccia, Campbell, and Thomas (2017)	✓			✓		✓
	Klapp, Erera, and Toriello (2016)				✓		✓
	Ulmer (2020)	✓					✓
	Klapp, Erera, and Toriello (2018)				✓		✓
	Ulmer and Thomas (2018)	✓				✓	✓
	RMDP, ACA	✓	✓	✓	✓	✓	✓

unique drop-off locations. When people are transported rather than goods, the DPDP is often referred to as the dial-a-ride problem (DARP). Psaraftis, Wen, and Kontovas (2016) provide a general overview of dynamic routing and include a categorization of dynamic DPDP. An earlier review of dynamic pickup and delivery can be found in Berbeglia, Cordeau, and Laporte (2010). In our review, we focus on only those problems labeled as dynamic and stochastic by Psaraftis, Wen, and Kontovas (2016). Like ours, these papers address the dynamic problem by incorporating probabilistic information into the solution approach.

The only other paper of which we are aware that addresses the dynamic and stochastic restaurant delivery problem is Reyes, Erera, and Savelbergh (2018). We note that Reyes, Erera, and Savelbergh (2018) became available after the completion of this paper. Reyes, Erera, and Savelbergh (2018) develop a rolling-horizon approach that incorporates postponement and bundling. This paper also considers postponement and bundling. Reyes, Erera, and Savelbergh (2018) also consider a benchmark that seeks to minimize “click-to-door” time, the amount of time that passes from when a customer orders to when the meal is delivered. The click-to-door objective is the same as minimizing delivery times. We test this objective as one of our benchmark policies. The papers differ most in our introduction of the buffer, the anticipatory element of our paper. We demonstrate in Section 6 that, although postponement and bundling are valuable in deriving good solutions, the buffer is the most important element of our solution approach. In essence, postponement and bundling do not explicitly account for future uncertainty as our parameterized buffer does.

There are a number of papers more generally addressing the DPDP. The applications are varied, and there does not seem to be a common approach. Mitrović-Minić and Laporte (2004) propose waiting strategies that seek to distribute time that is available for waiting across the time horizon. By waiting, Mitrović-Minić and Laporte (2004) implicitly recognize that there will be future requests. Fagerholt, Foss, and Horgen (2009) solve a DPDP related to air taxi service and make use of the waiting strategies proposed in Mitrović-Minić and Laporte (2004). In contrast to the work in this paper, Fagerholt, Foss, and Horgen (2009) determine pickup times for customers rather than face the uncertainty in ready times. Vonolfen and Affenzeller (2016) develop new waiting strategies based on historical request data and its relationship to the customer at which a vehicle is currently waiting. The construction of the waiting strategies in Vonolfen and Affenzeller (2016) explicitly captures probabilistic information about future requests. In this paper, we do incorporate a waiting

strategy, moving to the nearest “unoccupied” restaurant when there are no orders to assign to a vehicle. However, we must also account for the uncertainty of the waiting times and introduce the time buffer to do so.

Related to Mitrović-Minić and Laporte (2004) is Mitrović-Minić, Krishnamurti, and Laporte (2004). Mitrović-Minić, Krishnamurti, and Laporte (2004) not only incorporate waiting strategies but build routes using a strategy that discounts the combined cost and time consumption of customers who are served later in the horizon. Such a method is effective because it accounts for the fact that greater density generated from customers who request service in the future in fact reduces the actual marginal cost of serving these customers. The double-horizon method of Mitrović-Minić, Krishnamurti, and Laporte (2004) can, like the time buffers in our approach, be categorized as a cost function approximation (CFA). We do not label Mitrović-Minić, Krishnamurti, and Laporte (2004) as anticipating in Table 1 because the paper does not identify a learning method for the parameters of the CFA. The approaches differ in that the approach of Mitrović-Minić, Krishnamurti, and Laporte (2004) is designed to handle uncertainty resulting from future requests, and ours incorporates both the uncertainty of future requests as well as the uncertainty in ready times.

Cortés et al. (2009) use receding-horizon control to guide decision making in a DPDP. In the language of approximate dynamic programming, this approach can be categorized as a lookahead approach. See Powell (2011) for an overview of lookahead methods and approximate dynamic programming in general. Particularly, Cortés et al. (2009) look ahead using a set of scenarios of the future. Because of computational challenges, Cortés et al. (2009) are limited to a two-step lookahead horizon. In contrast, by using off-line simulation, we are able to determine the parameters of our policy looking across the entire horizon. In addition, because our approximations take place off-line, our method is scalable to much larger numbers of customers and vehicles. Muñoz Carpintero et al. (2015) propose a new evolutionary approach for solving the limited lookahead of Cortés et al. (2009) but still solve problems of only 11 vehicles. Cortés et al. (2009) address a dynamic DARP and, thus, passenger transportation. Therefore, their objective, as the objective in this paper, addresses the concerns of the customers. Núñez et al. (2014) adapt the method of Cortés et al. (2009) to multiple objectives that consider the trade-off between operator and user costs.

Cortés, Núñez, and Sáez (2008) extend the methodology of Cortés et al. (2009) to include stochastic travel times. As with the random ready times in this paper, stochastic travel times impact customer

service, and Cortés et al. (2009) seek to minimize the cost of waiting and travel times to customers. However, the resulting methodology is limited in its scalability. Sáez, Cortés, and Núñez (2008) introduce a more sophisticated procedure than that presented in Cortés et al. (2009) for developing predictions of future demand. As in Cortés, Núñez, and Sáez (2008), Schilde, Doerner, and Hartl (2011) consider a DPDP with stochastic travel times. Instead of the genetic algorithm for solving the hybrid predictive control problem as Cortés, Núñez, and Sáez (2008) do, Schilde, Doerner, and Hartl (2011) propose a variable neighborhood search to solve the probabilistic routing problems. Schilde, Doerner, and Hartl (2014) extends Schilde, Doerner, and Hartl (2011) to include time-dependent stochastic travel times. Both Schilde, Doerner, and Hartl (2014) and Schilde, Doerner, and Hartl (2011) use a method that requires real-time computation raising questions about its ability to scale, particularly when immediate decisions are needed as they are in the RMDP.

Hyytiä, Penttinen, and Sulonen (2012) consider a problem related to taxi dispatching. They use a heuristic based on M/M/1 queues to estimate the cost of an assignment of a vehicle. The method can be viewed as a value function approximation (VFA). VFA operates similarly to exact dynamic programming except that actions are selected via an approximate Bellman equation. This approximation is done on the second term of the Bellman equation, the cost to go. Our approach places the approximation on the first term of the Bellman equation, the current period reward, though we incorporate the value function by using a route-based Markov model. In addition, the VFA proposed by Hyytiä, Penttinen, and Sulonen (2012) considers each vehicle individually and not its future interactions. Our method of tuning our policy's parameter implicitly accounts for these interactions. Sayarshad and Chow (2015) extend Hyytiä, Penttinen, and Sulonen (2012) to consider pricing mechanisms to balance supply and demand.

Ghiani et al. (2009) study the DPDP in the context of courier dispatching in London. Their goal is to maximize customer service, which is expressed as a penalty on violations of the customers' delivery deadlines. In routing new customer requests, Ghiani et al. (2009) use sampling coupled with a routing algorithm to measure the impact of the new request in light of future customer requests. Although effective, the proposed method requires online computation and does not scale like our proposed approach.

Ghiani, Manni, and Romano (2018) seek a scalable method for a variant of the DPDP in which customers of different priorities are served over the problem horizon. Ghiani, Manni, and Romano (2018) propose a policy-function approximation approach (PFA). Their PFA is characterized by a policy with a tunable parameter.

The policy proposed by Ghiani, Manni, and Romano (2018) reserves a fraction of capacity so that it is available to serve dynamic requests from the highest priority class. They learn the best fraction to reserve using an off-line simulation. Conceptually, the PFA proposed in Ghiani, Manni, and Romano (2018) has a similar structure to the policy proposed in this paper. Because of the differences between the two problems, however, the PFA proposed in Ghiani, Manni, and Romano (2018) cannot be applied to the RMDP.

2.2. Same-Day Delivery

A recent variant of the DPDP is the SDDP. As in the dynamic DPDP, in the SDDP, requests arrive over time from geographically dispersed customers. In contrast to the DPDP, in the SDDP, all orders are picked up from the same pickup location. The authors are not aware of any papers in the SDDP literature that account for uncertainty in the ready time beyond the timing of customer requests nor an objective that seeks to minimize expected delays in delivery to customers. In the SDDP literature, only Azi, Gendreau, and Potvin (2012); Voccia, Campbell, and Thomas (2017); Ulmer (2020); and Ulmer and Thomas (2018) consider fleets of vehicles, and only Voccia, Campbell, and Thomas (2017) and Ulmer (2020) consider customer deadlines or time windows. Both Azi, Gendreau, and Potvin (2012) and Voccia, Campbell, and Thomas (2017) propose multiple-scenario planning approaches that require online computation that is too slow for real-time decision making given the number of vehicles and customers considered in this paper. Ulmer (2020) relies on a VFA approach that incorporates a lookup table. Although such approaches are effective, they are limited by dimensionality. The problem studied in this paper would require at least one dimension for each vehicle, which would quickly lead to memory issues. Ulmer and Thomas (2018) propose a scalable PFA, but the PFA is specific to the heterogeneous fleet studied in the paper and is not applicable to the problem studied here. Other SDDP literature includes Klapp, Erera, and Toriello (2016, 2018) and Ulmer, Thomas, and Mattfeld (2019), all of which focus on single-vehicle cases.

2.3. Further Related Literature

The authors are not aware of any literature that considers both stochastic customers and ready times. Although not summarized in Table 1, Arda et al. (2014) consider a problem in which customers are known, but the release times, analogous to ready times, are stochastic. In the paper and similar to the problem studied in this paper, the periods at which the known items become available for delivery is unknown. In contrast to the work in this paper, the problem studied in Arda et al. (2014) allows for

adequate time to make a decision, and the authors propose a number of online algorithms, algorithms for which much of the computation happens in real time. In addition, Archetti, Jabali, and Speranza (2015); Cattaruzza, Absi, and Feillet (2016); Reyes et al. (2018); Shelbourne, Battarra, and Potts (2017); and Archetti et al. (2018) study ready times in a deterministic context.

3. Model

In this section, we first present a formal description of the RMDP. We then give an example and model the problem as a route-based Markov decision process (MDP).

3.1. Problem Statement

The RMDP is characterized by a fleet of vehicles $\mathcal{V} = \{V_1, \dots, V_h\}$ that seeks to fulfill a random set of delivery orders $\mathcal{D} = \{D_1, \dots, D_m\}$ that arrive during the finite order horizon $T = [0, t_{\max}]$ from restaurants $\mathcal{R} = \{R_1, \dots, R_l\}$ located in a service area $\mathcal{G} = (\mathcal{N}, \mathcal{A})$. The service area \mathcal{G} consists of a set of nodes $\mathcal{N} = \{N_0, \dots, N_n\}$ and connections between the locations or arcs $\mathcal{A} = \mathcal{N}^2$. Each realized order D is associated with an order time $t^D \in T$, a delivery location $N^D \in \mathcal{N}$, and a pickup restaurant R^D . Each restaurant $R \in \mathcal{R}$ is characterized by a location $N^R \in \mathcal{N}$. Each arc (N_i, N_j) is associated with a travel time $d(N_i, N_j)$.

Each vehicle V_i begins service at an initial position $N_{\text{init}}^V \in \mathcal{N}$. Because drivers are independent contractors, each driver determines how assigned orders are sequenced and routed, for example, by using a tool such as Google maps to drive between locations. Each driver also decides when to reposition the vehicle and where to wait after completing a delivery and before another is assigned. However, we assume that these routing and sequencing procedures are known to the dispatchers. Because dispatcher and drivers communicate via mobile phones, diversions for new orders are not permitted while a driver is driving.

Orders occur according to a known stochastic process \mathcal{F}^D . A soft deadline $t^D + \bar{t}$ is associated with each order D . The time \bar{t} represents the amount of time that can pass between an order being placed and delivered without inconveniencing the customer. The time at which an order can be delivered depends in part on when the order is ready for pickup from the associated restaurant. The distribution of the ready time of order D depends on the restaurant R^D and is denoted \mathcal{F}^R . In this paper, we assume that the ready-time distribution is independent of the time of day and the restaurant's congestion level. However, the model can easily be adapted to account for those cases. The realized ready time $\rho(D)$ is revealed only if the assigned vehicle is located at the restaurant at the

time that the order is ready or after. We assume service times of t^R at a restaurant and t^s at a customer.

The dispatcher determines which orders are assigned to which vehicles. Assignments can be postponed. However, because, in reality, drivers are promised an order-specific reward for a delivery, once made, assignments cannot be altered. We assume that drivers accept all orders that are assigned during the order horizon. The assignment decisions combined with the drivers' sequencing and routing lead to a set of planned routes $\Theta = (\theta_1, \dots, \theta_h)$, one route for each vehicle. As noted earlier, the sequencing is determined by the drivers. In practice, drivers often use an insertion-type heuristic to make these routing decisions. We use similar methods in our computational experiments. Further, our proposed policy employs the drivers' routing procedures to determine the value of assigning an order to a particular driver.

Mathematically, a planned route is defined as a sequence of stops and the associated planned arrival time at each stop:

$$\theta = \left((N_1^\theta, a_1^\theta), (N_2^\theta, a_2^\theta), \dots, (N_o^\theta, a_o^\theta) \right).$$

The planned arrival times a reflect the dispatcher's assumptions about when a vehicle will arrive at each location. These planned times change as ready times and new customer requests are realized. Because the actual arrival times are impacted by stochasticity in requests and ready times, these planned times are not necessarily the expected time of arrival to the customer. The first entry of a planned route represents the next location a vehicle will visit and the arrival time at this location. Because the vehicle is en route and is not diverted, the arrival time to the first entry is known.

A delivery is realized when the vehicle has picked up the order at the restaurant, arrived at the customer, and completed the delivery. The time α_{real}^D represents the random time of delivery D . In case the realized arrival time is later than the deadline, the delivery is delayed. The dispatcher seeks to minimize the expected sum of the delay of deliveries:

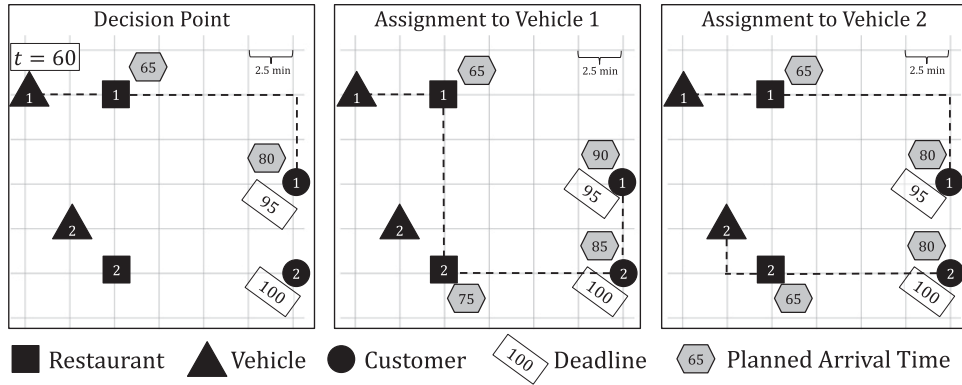
$$\min \mathbb{E} \left[\sum_{D \in \mathcal{D}} \max\{0, \alpha_{\text{real}}^D - (t^D + \bar{t})\} \right].$$

3.2. Example

In the following, we present a simple example of the RMDP. We use this example in our description of the MDP and in motivating our solution method.

The example is depicted in Figure 1. The figure shows two restaurants (squares), two vehicles (triangles), and two customers (circles). The deadlines at the customers are displayed in the rectangles.

Figure 1. Example for the MDP Model



The planned arrival times at restaurants and customers are displayed in the hexagons. We note that the departure times from the restaurants may not necessarily be the same as the arrival times because vehicles may arrive early at restaurants. In this example, we assume a Manhattan-style grid with travel times of 2.5 minutes per segment, and we ignore service times. The mean of the ready-time distribution at the restaurants is 10 minutes, and $\bar{t} = 40$ minutes. On the left side of Figure 1, a decision point at time $t = 60$ and the state at the decision point are shown. At this decision point, customer D_1 has already ordered from restaurant 1 at time $t^{D_1} = 55$. Customer 2 ordered from restaurant 2 at time $t^{D_2} = 60$, inducing the decision point. The deadlines for the orders are, therefore, 95 and 100, respectively. Customer 1 is already assigned to vehicle 1 and routed with respect to the routing routine. The planned route for vehicle 1 is $((R_1, 65), (D_1, 80))$ with planned arrival times of 65 and 80. The computation of these planned arrival times is discussed in Section 4. The second vehicle is idling at location N^{V_1} . The planned route for this vehicle is $((N^{V_1}, 60))$, meaning that vehicle 2 is currently located at location N^{V_1} . The current assumed delay of the plans is zero because the arrival time at customer 1 is earlier than the deadline. At this decision point, the decision is to determine the assignment of customer 2. This customer can be assigned to vehicle 1 or vehicle 2, or the assignment can be postponed for an amount of time.

In the following, we describe the two potential assignments for customer 2: the assignment to vehicle 1 and the assignment to vehicle 2. The resulting routing plans are depicted in the center and on the right side of Figure 1, respectively. The middle plan reflects the assignment of customer 2 to vehicle 1. The updated plan for vehicle 1 is then

$$((R_1, 65), (R_2, 75), (D_2, 85), (D_1, 90)).$$

Again, the assumed delay of the planned route is zero.

The assignment of customer 2 to vehicle 2 is shown on the right side of Figure 1. The resulting routing plans are $((R_1, 65), (D_1, 80))$ for vehicle 1 and $((R_2, 65), (D_2, 80))$ for vehicle 2. Having arrived at restaurant 2 at time 65, an expected ready time at restaurant 2 of 10 minutes, and 25 minutes of travel time from restaurant 2 to customer 2, the planned arrival time at customer 2 is time 80.

Once an assignment is made, the drivers proceed with their plan. Assume the food for customer 1 at restaurant 1 requires 20 minutes of preparation and is completed at time 75. Thus, vehicle 1 needs to wait for 10 additional minutes at restaurant 1. If customer 2 is assigned to vehicle 1 (the middle case), this leads to realized arrival times of $\hat{a}_{\text{real}}^{D_1} = 100$ at customer 1 and $\hat{a}_{\text{real}}^{D_2} = 95$ at customer 2. The realized delay at customer 1 is 5 minutes. If customer 2 is assigned to vehicle 2, the realized arrival times are 90 at customer 1 and 80 at customer 2. In this case, all customers are served without delay.

3.3. Route-Based Markov Decision Process

In this section, we present an MDP model for the RMDP. We draw on the route-based MDP formulation introduced in Ulmer et al. (2017). The route-based formulation augments the decision space of a conventional MDP to determine and update a route plan. Consequently, we also augment the conventional MDP state space to include the planned routes. The advantage of this route-based modeling framework for the RMDP is that, in the RMDP, the cost associated with a particular decision is not known until a delivery is actually made. The proposed formulation associates a “planned” cost with each decision at the time the decision is selected. These costs represent the deterministic part of the costs based on the planned arrival times. They are combined with “stochastic” costs resulting from the difference in planned and realized arrival times because of the uncertainty in the

ready times and the arrival of new orders. In addition, in Section 4, we propose a solution approach that operates on planned routes. Given this approach, the route-based MDP formulation provides a connection between the problem and the solution method in the same way that set-cover formulations provide a connection for column generation-based approaches to the vehicle routing problem.

In the following, we define the MDP components: decision point, decision state, decisions, stochastic transitions, and costs.

3.3.1. Decision Points. In the RMDP, decision point k occurs as a result of one of two events. First, a decision point occurs when a new customer D_k^{new} requests service at time t_k . Second, a decision point can be “self-imposed.” This happens when an order is postponed at decision point $k - 1$ for the next δ units of time. In the case that no customer requests service in the next δ time units after t_{k-1} , decision point k is then at time $t_k = t_{k-1} + \delta$. The initial decision point occurs when the first customer orders. The total number of decision points K is random, and the process terminates when all orders are served and the order horizon has ended, $t_K \geq t_{\max}$.

3.3.2. Decision States. Decision states represent the information needed to make a decision at a decision point. A decision state S_k consists of the point of time t_k of the current decision point k and the set of orders \mathcal{D}_k , including the new order D_k^{new} if applicable. Because we are formulating a route-based MDP, the decision state also includes the planned routes Θ_k .

Each order $D \in \mathcal{D}_k$ is a vector, $D = (t^D, R^D, V^D, L^D)$, where t^D is the time of request D , R^D the restaurant associated with the request, V^D the vehicle assigned to the request, and L^D the loading status of the request. Parameter V^D is a number between zero and $|V|$ and indicates the vehicle to which order D is assigned. When $V^D = 0$, the order has not yet been assigned. Parameter $L^D = \{0, 1\}$ indicates whether the order is already loaded on the vehicle ($L^D = 1$) or not ($L^D = 0$).

3.3.3. Decisions and Deterministic Costs. A decision x at decision point k assigns orders to vehicles by changing the set of orders \mathcal{D}_k to \mathcal{D}_k^x . A decision x , therefore, results in an update of the routing plan Θ_k to Θ_k^x . The resulting state is called the postdecision state and is denoted S_k^x . If the following conditions hold, a decision is feasible:

1. The arrival time of the first entry of each route $\theta \in \Theta_k$ remains the same.
2. The value V^D stays unaltered for all orders D with $V^D > 0$ (assignments are permanent).
3. The sequencing in Θ_k^x reflects the driver’s routing routine.

Postponements are possible for orders D with $V^D = 0$ for an order D . In case of a postponement, the decision further contains an amount of time δ until the postponed orders are reevaluated for assignment. We also note that this definition of the decisions supports bundling, and the decisions do not exclude the possibility of routing multiple restaurants one after the other and then serving the associated customers.

We note that the definition of a planned route allows arbitrary assumptions about the planned arrival times for all routed customers except the first one on each route. In our method, we base the arrival times on the expected ready, travel, and service times given the information at decision k . We describe our procedure in more detail in Section 4.

Using the route-based model formulation, we can break the costs of a particular action into two parts: a deterministic and a stochastic part. The deterministic part of the costs C^d reflects the marginal change in planned delay resulting from a decision. These costs may, therefore, be negative. We first define a delay function that operates on decision states and postdecision states. For the decision state, we have, for any k ,

$$\Delta(S_k) = \sum_{\theta \in \Theta_k} \sum_{D \in \theta} \max\{0, a^D - (t^D + \bar{t})\},$$

where a^D is the planned arrival time associated with customer D . The function for postdecision states is analogous and given as

$$\Delta(S_k^x) = \sum_{\theta \in \Theta_k^x} \sum_{D \in \theta} \max\{0, a^D - (t^D + \bar{t})\},$$

and the costs are the difference between updated planned delay $\Delta(S_k)$ and currently planned delay $\Delta(S_k^x)$ and are calculated as

$$C^d(S_k, x) = \Delta(S_k^x) - \Delta(S_k). \quad (1)$$

3.3.4. Transition and Stochastic Costs. A new decision point is triggered at time t_{k+1} when either a new customer D_{k+1}^{new} orders, when the time for postponement δ has passed, or when no orders are unassigned and the request arrival phase ended. The latter leads to the termination state of the process.

In either case, we observe the k^{th} realization of the exogenous information $\omega_k \in \Omega_k$. The exogenous information contains two parts $\omega_k = (\omega_k^R, \omega_k^D)$ with ω_k^R being the stochastic information about the potentially observed ready times and ω_k^D about the potential new customer order. We note that both ω_k^R and ω_k^D may be empty sets.

A stochastic realization also leads to the next decision state $S_{k+1} = (S_k^x, \omega_k)$ or the termination state. The transition changes the postdecision state as follows:

- The new time is $t_{k+1} = t_k + \delta$ in case of no request or the time t_{k+1}^{new} the new order was made.
- The combination of realization $\hat{\omega}_k$ with the routes Θ_k^x reveals a set of stops made between t_k and t_{k+1} as well as the customers served in that time span. We denote the set of served customers \mathcal{D}_k^ω .
- The set of yet-to-be-served orders is $\mathcal{D}_{k+1} = (\mathcal{D}_k \cup \omega_k^D) \setminus \mathcal{D}_k^\omega$ with ω_k^D either being the empty set or containing $\mathcal{D}_{k+1}^{\text{new}}$.
- In the case that an order for customer D was picked up, the loading status of this customer is set to loaded ($L^D = 1$).
- The routes Θ_k^x are updated by removing all realized stops and by updating the arrival time of the first entry of each route $\theta \in \Theta_{k+1}$ with respect to the ready times observed in ω_k^R . All other arrival times remain the same as in the previous planned routes Θ_k^x (they might be updated with the next decision though).

The transition also results in additional costs that were not planned. The reason for the costs are two-fold. First, we may have delivered the orders for customers in \mathcal{D}_k^ω at different times than planned. Second, we observed changes in the arrival time of the first entry of each route $\theta \in \Theta_{k+1}$. If this first entry in the route is a customer, we serve the customer at a potentially different time than planned. The stochastic cost reflects all differences in planned and realized arrival times for the served customers and the customers that are visited with the next stops of the vehicles. We denote this set of affected customers $\mathcal{D}_k^{\omega, \Theta}$. We denote the realized arrival time for each customer $D \in \mathcal{D}_k^{\omega, \Theta}$ as $\alpha_{\text{real}}^D(\omega_k)$. Then, the realized stochastic cost is defined as

$$C^s(S_k^x, \omega_k) = \sum_{D \in \mathcal{D}_k^{\omega, \Theta}} \left[\max(\alpha_{\text{real}}^D(\omega_k) - (t^D + \bar{t}), 0) - \max(a^D - (t^D + \bar{t}), 0) \right].$$

We note that the costs C^s depend on only the postdecision state and the realization of the arrival times and do not depend on the decisions that are made in next predecision state S_{k+1} . To account for these realized stochastic costs, we add these costs to the deterministic costs described in Equation (1).

3.3.5. Objective. A solution of the MDP is a policy $\pi \in \Pi$. A policy is a sequence of decision rules $\pi = (X_0^\pi(S_0), \dots, X_K^\pi(S_K))$ mapping states to actions. The objective for the MDP is to determine an optimal policy π^* minimizing the expected costs given by

$$\min_{\pi \in \Pi} \mathbb{E} \left[\sum_{k=0}^K C^d(S_k, X_k^\pi(S_k)) + C^s(S_k^{X_k^\pi(S_k)}, \omega_k) \middle| S_0 \right]. \quad (2)$$

3.4. Example Revisited

To conclude this section, we revisit the example and embed it in the MDP notation. The state S_k occurs at time $t_k = 60$, and the planned routes are $\Theta_k = (((R_1, 65), (D_1, 80)), (N^{V_2}, 65))$ with N^{V_2} being the current location of vehicle 2. The customers are represented by $\mathcal{D}_k = \{(N^{D_1}, 55, 1, 0), (N^{D_2}, 60, 0, 0)\}$.

The decision x is the assignment of customer 2 to vehicle 1. The updated routing plan is

$$\Theta_k^x = (((R_1, 65), (R_2, 75), (D_2, 85), (D_1, 90)), (N^{V_2}, 65)).$$

Following the decision, the customers are $\mathcal{D}_k^x = \{(N^{D_1}, 55, 1, 0), (N^{D_2}, 60, 1, 0)\}$. The decision leads to deterministic costs of $C^d(S_k, \Theta_k, \Theta_k^x) = \Delta(S_k, \Theta_k^x) - \Delta(S_k, \Theta_k) = 0 - 0 = 0$.

The stochastic transition is induced by the next order at time $t = 105$. At that time, the realized ready time for D_1 is 75. The realized arrival times are $\hat{\alpha}_{\text{real}}^{D_1} = 100$ and $\hat{\alpha}_{\text{real}}^{D_2} = 95$. The stochastic costs are, therefore, $C^s(S_k, \Theta_k^x, \omega) = \max(\hat{\alpha}_{\text{real}}^{D_1} - 95, 0) - \max(a^{D_1} - 95, 0) + \max(\hat{\alpha}_{\text{real}}^{D_2} - 100, 0) - \max(a^{D_2} - 100, 0) = 5 + 0 = 5$.

4. Solution Approach

In this section, we present our solution method, the ACA. We first motivate the components of the methods. We then define the individual components and finally give the algorithmic details.

4.1. Motivation

The RMDP is plagued by all three “curses of dimensionality.” First, with a fleet of vehicles and many potential unassigned requests at any given time, the state space is characterized by multidimensional vectors, grows combinatorically in those dimensions, and is thus extremely large in real-world situations. Second, the decision space requires the assignment of requests, a problem that is complex in static and deterministic settings and here exacerbated by the option to postpone assignments. Finally, uncertainty is present in two dimensions. The requests are uncertain as is the ready time associated with the request. Except in stylized cases, convolutions of these distributions cannot be analytically determined.

As a result, we are not able to obtain optimal policies for real world-size problems. To overcome this challenge, we introduce a heuristic assignment policy for the RMDP that allows us to address the uncertainty while maintaining computational tractability in the face of the large state and decision spaces.

The method must account for random arrival of new customer requests and the uncertain ready times at the restaurants. Our heuristic addresses both of these uncertainties by postponing the assignments of orders and by buffering the planned arrival times of the customers. The heuristic further employs a

runtime-efficient assignment procedure to account for the limited availability of calculation time. Before we present the algorithmic details of our policy, we briefly motivate and illustrate the functionality of our two main components: postponement and time buffer.

4.1.1. Postponement. We implement a postponement policy that postpones a set of “not yet important” orders. To this end, we plan tentative assignments of all customers, but postpone assignments that are not related to the next stop of a route. Customers are officially assigned only when their restaurants are the next stop on a driver’s route. We expect postponements will allow the accumulation and potential bundling of requests and, thus, create greater flexibility in later assignment decisions. We expect postponements to be particularly important when the gain in information over time is high. Such a gain is particularly likely when the workload is high as a result of many customer requests. In these cases, we are able to obtain significantly more efficient assignments for the postponed and new requests. Further, we expect postponements to be beneficial when the variance in the ready times is high. In these cases, delays in ready times likely render previously efficient assignments inefficient, meaning that there is value in waiting to make assignments.

4.1.2. Time Buffer. The time buffer is a cost that is added to the planned arrival times at the customers. The idea behind the time buffer is twofold. The first and most intuitive reason for the time buffer is to hedge against the uncertainty in ready times and the random arrival of new customers. The buffer discourages decisions that plan for a driver arriving at a customer close to the customer’s delivery deadline. In essence, the buffer is available to absorb uncertainty resulting from either the ready times or new customer arrivals that need to be inserted in routes. Thus, we expect to benefit from a large time buffer when there is high volatility in the ready times and when arrival rates are high.

Another role of the time buffer is more subtle. Different buffers change the utilization of our fleet. For example, because orders seem less urgent, a small buffer enables order bundling. Bundling creates flexibility by leaving other vehicles available to serve potential future customers. With an increasing buffer and, thus, more urgency to each customer’s delivery, the bundling operations decrease and policies tend to assign the vehicle that can serve the customer fastest. We expect that the choice of a suitable buffer depends on the workload of the vehicles. For low workload and sufficient resources, the buffer is relatively large because sending the “fastest” vehicle does not impede service of future customers and, thus, the focus can be

on absorbing delays resulting from random ready times. With increasing workload, the buffer decreases because it balances the trade-off between hedging against delay and future flexibility available from bundling.

We next describe the policy in detail. We first present the assignment heuristic. Then, we describe how we implement the postponement strategy and how we integrate the buffer. Finally, we present the algorithmic procedure summarizing the three features of our method.

4.2. Assignment Heuristic

For the problem under consideration, decisions are made only about the assignment of unassigned orders to drivers. The route plan is determined by the drivers. Thus, each assignment results in a specific unalterable route plan. Given a set of planned routes and a set of unassigned orders, the procedure determines the “best” vehicle assignment for each of the unassigned orders.

Given o unassigned orders and h vehicles, the number of potential assignments is h^o without considering postponements. Thus, we reduce the number of assignment solutions under consideration.

First, we enumerate all potential ordered sequences of unassigned orders. Given o unassigned orders, this leads to a set of $o!$ sequences. With four unassigned customers, this results in 24 sequences of four customers. Then, for each of the enumerated sequences and, in turn, each order in a sequence, we determine the vehicle for the assignment. The potential assignment of an order to a vehicle impacts the planned delay of all orders assigned to the vehicle. The procedure assigns the order to the vehicle that minimizes the increase in planned delay. As noted previously, we assume that we know each driver’s routing routine and are, thus, able to evaluate the delay caused by an assignment. In case there are several such vehicles minimizing delay, ties are broken by choosing the vehicle for which the average slack per customer, the average difference between planned arrival times and customer deadlines, is highest. When the order is assigned, the procedure continues with the next order in the sequence currently being considered. When all sequences have been evaluated, the sequence is selected, minimizing the marginal change of planned delay. With this procedure, instead of 50,625 possible assignments, we consider only $24 \times 4 \times 15 = 1,440$ different assignment solutions with 15 vehicles and four unassigned customers.

With respect to the route-based MDP, this procedure reduces the decision space and selects the sequence and the associated assignments leading to the overall minimal change in planned delay and, thus, the deterministic costs $C^d(S_k, x) = \Delta(S_k^x) - \Delta(S_k)$.

4.3. Postponement Strategy

One way to account for uncertainty in customer orders and ready times is to postpone the assignment of customers. A postponement of assignments allows us to learn about new orders and realized ready times before we commit to a decision. Thus, we expect that postponing orders is particularly suitable if the gain in information is high, which happens when arrival rates are high or when ready times are highly variable.

We determine the postponements as follows. Because some orders may be most efficiently served by immediate assignment, we postpone only orders that are not related to the next stop of a planned route. This means that we postpone orders only for which an assignment would not impact the next position of the assigned vehicle's current route. Thus, we do not postpone orders for which the updated route results in the vehicle driving to the restaurant associated with the order. All other orders can be postponed without any immediate consequences in the route and may be later integrated in the route again. To allow such a potential integration, we save the route plan that we would choose had we not allowed postponement. In case no better assignment solution emerges, we are then able to execute the previously planned assignments.

The number of postponements impacts the computational effort for the aforementioned assignment procedure. Computation time increases as the number of postponed orders increases. Because our computational study requires a large number of simulations, we limit the number of possible postponements to a fixed value p_{\max} . As a result, only the first p_{\max} eligible orders of the sequence can be postponed. However, we show in our experiments that increasing the number of postponements does not add significant benefit. We allow orders to be postponed at most t_{\max}^p minutes.

4.4. Time Buffer

Decision making based on only the deterministic costs C^d ignores the impact of the random ready times and potential future orders. As shown in the example in Section 3.2, even though the deterministic costs might be zero for all decisions, different decisions lead to different stochastic costs. These stochastic costs result from ready times being later than planned and from later customer requests being inserted into the routes. To account for these uncertain outcomes, we introduce a time buffer as a type of CFA. Powell (2019, p. 13) notes, "CFAs are widely used for solving large scale problems such as scheduling an airline or planning a supply chain. For example, we might introduce slack into a scheduling problem, or buffer stocks for an inventory problem." In analogy, the base of our CFA is a temporal buffer b .

This buffer is used as a penalty cost term to avoid "risky" arrivals that are close to a customer's deadline. Specifically, we artificially increase the planned arrival times at customers by the amount of the buffer. For a given buffer b , the new planned delay $\Delta^b(S_k^x)$ for state S_k and decision x is then defined as

$$\Delta^b(S_k^x) = \sum_{\theta \in \Theta_k^x} \sum_{D \in \theta} \max\{0, (a^D + b) - (t^D + \bar{t})\}.$$

Deliveries with a planned arrival time $(a^D + b)$ contribute to the objective only if $a^D + b > t^D + \bar{t}$. A buffer greater than zero, thus, makes it more likely a delivery contributes, albeit artificially, to the objective function because $a^D + b$ is more likely to be greater than $t^D + \bar{t}$. A buffer of zero leads to the original delay function $\Delta(S_k^x)$, which depends on only deterministic costs. Further, a buffer of \bar{t} leads to a planned delay of

$$\Delta^{\bar{t}}(S_k^x) = \sum_{\theta \in \Theta_k^x} \sum_{D \in \theta} \max\left\{0, \underbrace{(a^D + \bar{t}) - (t^D + \bar{t})}_{=a^D - t^D}\right\}. \quad (3)$$

Delay as computed in Equation (3) results in sending the fastest vehicle to an order. We use this buffer as one of our benchmark policies and to reflect current practice.

Revisiting the example in Section 3.2, a buffer of $b = 10$ minutes results in a planned delay of $\Delta^b(S_k^x) = 5$ minutes and an assignment of customer 2 to vehicle 2, whereas the customer was originally assigned to vehicle 1. Customer 2 is assigned to vehicle 2 for buffers $b > 5$. With the transition described in the example, this assignment accounts for the possibility of the delay and leads to the best realized costs.

We tune the buffer using sample average approximation (SAA) on the restricted policy class. See Fu (2015) for an overview of SAA. Details of our procedure can be found in Section 5.3.

4.5. Algorithmic Procedure

In this section, we summarize the algorithmic procedure for executing our solution approach. Algorithm 1 presents the procedure. In a decision state, the algorithm determines the assignments and postponements based on the three previously introduced features.

The input parameters of the algorithm are the state S with point in time t , the current route plan Θ , and the unassigned orders \mathcal{D}^o . Other input parameters are the buffer b , maximum number of postponements p_{\max} , and the maximum amount of time that an assignment can be postponed t_{\max}^p . Output is the decision x that includes the updated route plan Θ^x and the set of postponed customers \mathcal{P}^x . This set is then used in the MDP to update the set of orders \mathcal{D}^x . Essentially, the

algorithm iterates through all ordered sets $\hat{\mathcal{D}}$ of \mathcal{D}^o and selects the one resulting in minimal delay. The algorithm considers the time buffer in the calculation of the delay. In the case of several candidates with similar delay, the one with the largest average slack per customer is selected. For each candidate route plan, the algorithm selects orders eligible for postponement. Eventually, it removes these orders from the route plan and returns both the updated route plan and the postponed orders.

The algorithm makes use of the following functions:

FindVehicle($\hat{\Theta}, D, b$): This function returns the vehicle of route plan $\hat{\Theta}$, where order D can be inserted with minimal increase in delay based on the drivers' routing routine. This function integrates the buffer b in the calculation of the delay.

AssignOrder($\hat{\Theta}, D, V$): Given a route plan, an order, and a vehicle, the function returns an updated route plan with order D integrated in the route of vehicle V based on the driver's routing routine. This function accounts for the possibility of bundling.

Postponement($\hat{\mathcal{P}}, \hat{\Theta}, D, p_{\max}, t_{\max}^p$): Based on the set of already postponed customers $\hat{\mathcal{P}}$, the route plan, and the two postponement parameters p_{\max} and t_{\max}^p , this function returns a Boolean variable indicating whether this customer can be postponed.

Slack($S, \hat{\Theta}$): For every route plan $\hat{\Theta}$, the function calculates the sum of differences between arrival time a^D and deadline over all orders: $\max\{0, (t^D + \bar{t}) - (a^D + b)\}$.

Remove(Θ^x, \mathcal{P}^x): This function removes the postponed orders and the according restaurant visits from the planned routes.

Algorithm 1 (Algorithm for the RMDP)

Input: State S , time t , route plan Θ , unassigned orders \mathcal{D}^o , buffer b , maximal number of postponements p_{\max} , maximum time allowed for postponement t_{\max}^p

Output: Route plan Θ^x , postponed orders \mathcal{P}^x

```

1 // Initialization
2  $x \leftarrow \emptyset$  // Best decision
3  $\text{delay} \leftarrow \text{bigM}$  // Delay
4  $\text{slack} \leftarrow 0$  // Slack
5 // Assignment Procedure
6 forall  $\hat{\mathcal{D}}$  ordered set of  $\mathcal{D}^o$  // All potential sequences
7 do
8    $\hat{\Theta} \leftarrow \Theta$  // Candidate route plan
9    $\hat{\mathcal{P}} \leftarrow \emptyset$  // Set of postponements
10  forall  $D \in \hat{\mathcal{D}}$  // All orders in sequence
11  do
12     $V \leftarrow \text{FindVehicle}(\hat{\Theta}, D, b)$  // Find best vehicle
13     $\hat{\Theta} \leftarrow \text{AssignOrder}(\hat{\Theta}, D, V)$  // Assign order

```

```

14   if (Postponement( $\hat{\mathcal{P}}, \hat{\Theta}, D, p_{\max}, t_{\max}^p$ ) == true)
15     // If postponement possible
16     then
17        $\hat{\mathcal{P}} \leftarrow \hat{\mathcal{P}} \cup \{D\}$  // Postpone order
18        $\hat{x} \leftarrow (\hat{\Theta}, \hat{\mathcal{P}})$  // Create candidate decision
19     end
20   if ( $\Delta(S^{\hat{x}}) < \text{delay} \vee ((\Delta(S^{\hat{x}}) == \text{delay}) \wedge (\text{Slack}(S, \hat{\Theta}) < \text{slack}))$ ) // If best route plan
21   then
22      $x \leftarrow \hat{x}$  // Update decision
23      $\text{delay} \leftarrow \Delta(S^{\hat{x}})$  // Update delay
24      $\text{slack} \leftarrow \text{Slack}(S, \hat{\Theta})$  // Update slack
25   end
26  $\Theta^x \leftarrow \text{Remove}(\Theta^x, \mathcal{P}^x)$  // Removed postponed orders and restaurants
27 return Route plan  $\Theta^x$ , Postponed orders  $\mathcal{P}^x$ 

```

5. Experimental Design and Implementation

In this section, we present the experimental design and introduce the benchmarks that we use to demonstrate the performance of ACA. We also describe the implementation that we use for our computational tests.

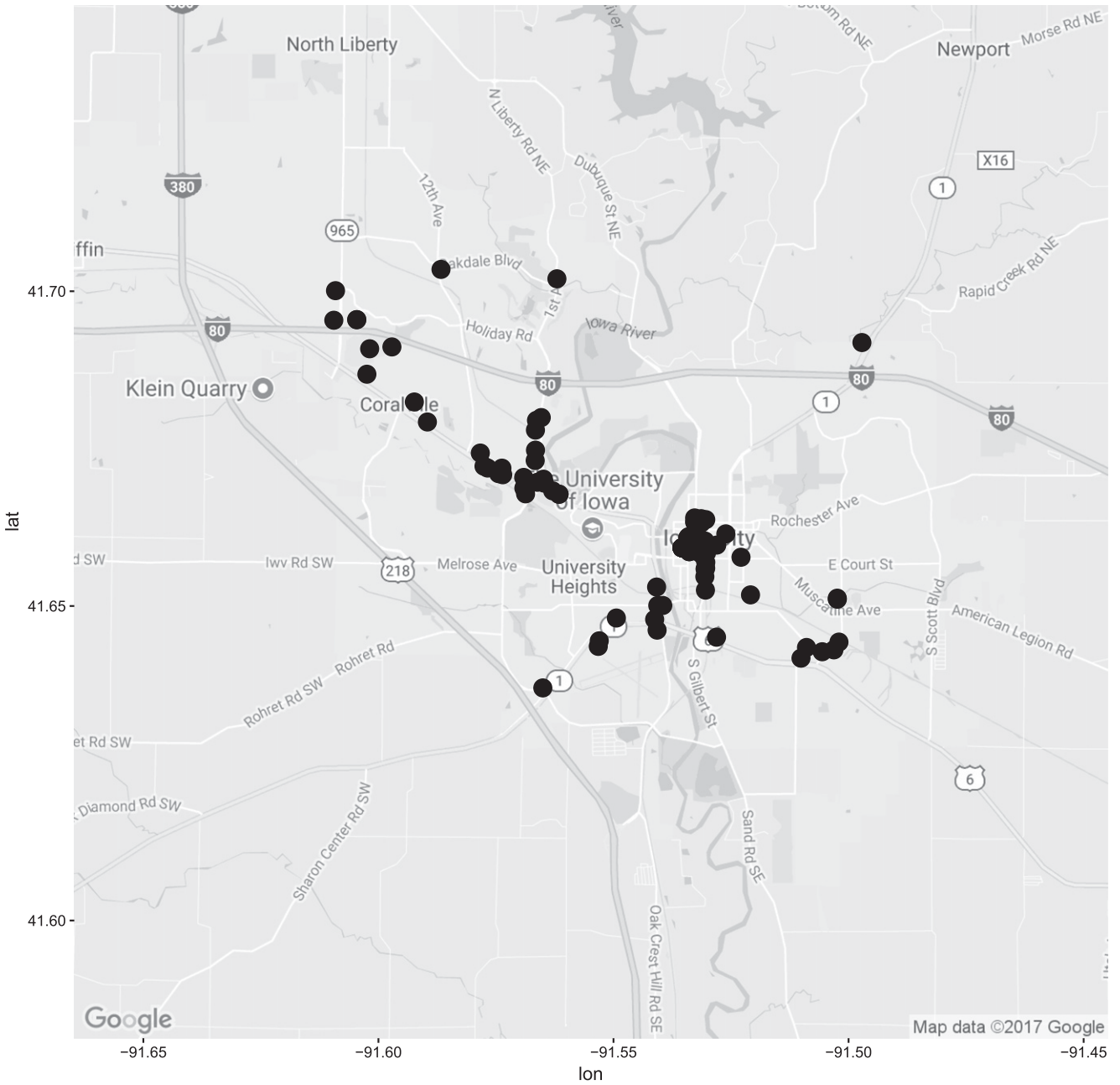
5.1. Instance Settings

Our experiments represent the delivery of restaurant meals in Iowa City, a city whose metropolitan statistical area had an estimated population of 168,828 people in 2016 (U.S. Census Bureau 2017). Iowa City is representative of a medium-sized city where such meal delivery services are used or a delivery zone within a larger city.

For each instance, the locations of the customers are randomly chosen from 32,000 potential customer locations in the Iowa City metropolitan area. These locations were pulled from the Geographic Information System Division of Johnson County, the Iowa county in which Iowa City is located. We assume that arrival of customer orders follow a homogeneous Poisson process. Conversations with local providers suggest that this assumption matches the request pattern of Iowa City. To compute the delivery deadlines, we set the value of $\bar{t} = 40$ minutes for every order.

We consider 110 restaurants based on the list of restaurants available from a meal delivery company operating in Iowa City in summer 2016. The locations of the restaurants are shown in Figure 2. Most of the restaurants are either in downtown Iowa City or in the suburb of Coralville. We observe several restaurant

Figure 2. Restaurant Locations



clusters that offer potential for bundling orders and also single restaurants distributed within the area. The time to prepare meals at each restaurant is assumed to be gamma distributed with a mean of 10 minutes and different variances. We choose the gamma distribution because it is skewed and has a long tail. This form reflects that orders are not usually ready significantly earlier than expected. Notably, for most meals, there is some assembly and cooking time that cannot be shortened. However, for various reasons, including other existing orders and staff shortages, some orders are ready significantly late.

Unless stated otherwise, we consider a fleet of 15 vehicles originating in different parts of the service area. We allow orders to arrive during the first seven hours of the horizon, $T = [0, 420]$ minutes. This usually leads to working hours of less than eight hours for the drivers of these vehicles and would correspond to a working day of noon to 8 p.m. The service time at a customer and at a restaurant, once the food is ready, is two minutes.

We assume drivers deploy the following routing procedure. Drivers follow a mobile navigation device to determine the best paths between customers and/

or restaurants. Because we have so many potential pairwise paths, it is not possible to find these travel times directly because of restrictions associated with the commercial mapping services (such as Google-Maps). Thus, we approximate the travel time of these paths by multiplying the Euclidean distance by a factor of 1.4 and assuming the travel speed over the resulting distances is 40 kilometers per hour. Boscoe, Henry, and Zdeb (2012) demonstrate that multiplying Euclidean distances by a factor of 1.4 closely approximates the relationship between Euclidean and street distances. If the driver has at least one unserved order when the driver receives a new order, we assume that the driver determines the best sequence to serve the bundle using a generalization of the cheapest insertion (CI) method by Rosenkrantz, Stearns, and Lewis (1974). Given the current planned sequence and the new order, the procedure first checks whether the restaurant for the new order is already part of the planned sequence. Otherwise, the procedure inserts the restaurant via CI. Then, the customer is inserted via CI after the restaurant's position. Once a vehicle has served its last assigned order, the driver repositions to the closest empty restaurant. A restaurant is called "empty" if no other driver is currently idling there or on the way to that restaurant.

To analyze the impact of the workload per vehicle, we vary the number of orders per hour per vehicle in the set 1.5, 2.0, 2.5. This leads to instance settings with a light workload (180 expected total orders), a normal workload (240), and a heavy workload (300). Second, to account for stochastic meal preparation times, we vary the coefficient of variation (COV) associated with ready times between 0.0, 0.1, ..., 0.6. A COV of zero is equivalent to deterministic meal preparation times. A COV of 0.6 leads to highly volatile ready-time realizations.

In total, we have 21 different instance settings. We evaluate ACA and four benchmarks using 2,000 instance realizations per instance setting. For variance reduction, we use the same 2,000 realizations across ACA and the benchmarks. This creates a total of nearly 210,000 (21 instance settings \times 2,000 realizations \times 5 policies) evaluation runs with about 70 million decision points.

5.2. Benchmarks

The proposed policy, ACA, draws on both postponements and a time buffer. We compare the policy to a series of benchmarks that are variations of ACA. In total, we consider four benchmark policies:

No buffer, no postponement: To analyze the impact of the combination of the postponement and the buffer, we test one policy with no buffer and no postponement of orders. This policy operates on only

the deterministic cost function and does not include the buffer. We refer to this policy as *Deterministic*.

Buffer, no postponement: To isolate the impact of the buffer, we consider a policy in which we do not allow postponement but for which we do include the time buffer. We refer to the policy using a time buffer with no postponement as *Buffer*. This policy is implemented by setting $p_{\max} = 0$. We choose the best buffer for Buffer in a manner analogous to the procedure we use for ACA. Thus, the time buffer may differ between the two policies.

Postponement, no buffer: To isolate the impact of postponement, we also consider a policy that considers postponement, but no time buffer. We refer to this policy as *Postpone*.

Current practice: We also consider a policy with a buffer of 40 minutes and call the policy *Fastest*. With a buffer of 40 minutes, the policy seeks to send the vehicle that can serve the order fastest based on current information (and delivery times for the remaining orders). In effect, this policy seeks to minimize, albeit myopically, the travel time associated with serving orders. This policy represents the current practice of some restaurant meal providers. We use this policy as the base policy to which we compare all other policies.

5.3. Implementation Details

In this section, we describe the implementation details of ACA and the benchmark policies. For both ACA and Buffer, we tune the buffer using sample average approximation on the restricted policy class. Let π^b be the described policy (ACA or Buffer) with buffer b . We define policies $b = 0, 2, \dots, \bar{t}$ minutes for each instance setting. For each instance setting, we run 2,000 simulated realizations $\Omega^1, \dots, \Omega^{2,000}$ and select the buffer minimizing the average delay. We note that these tuning realizations are different than the 2,000 realizations used for evaluation. Let $\mathcal{C}(\Omega^i, \pi^b)$ be the accumulated delay when applying policy π^b to realization Ω^i . Mathematically, we choose

$$\pi^{\text{best}} = \underset{\{\pi^b: b=0, 2, \dots, \bar{t}\}}{\operatorname{argmin}} \left\{ \frac{1}{2,000} \sum_{i=1}^{2,000} \mathcal{C}(\Omega^i, \pi^b) \right\}.$$

We set p_{\max} and t_{\max}^p based on preliminary tests. For all instances, we set the maximum time of postponement to $t_{\max}^p = 30$ minutes. We note that the policies are nearly invariant for $t_{\max}^p > 15$. For instances with light and normal workload, we set the number of postponements to $p_{\max} = 3$. For instances with heavy workload, we set $p_{\max} = 2$ because of computational limitations.

The algorithms are implemented in Java. We run the tests on Windows Server 2008 R2, 64 bit with Intel-Xeon

E7-4830@2.13GHz, 64 cores, and 128 GB RAM. Our implementation uses only a single core for each simulation run. The runtimes per decision point are generally less than a second, thus making the method capable of use in a real-time decision-making environment.

6. Policy Performance

This section analyzes the performance of ACA and compares its performance to the benchmark policies. We first compare the solution quality of the policies and then discuss the impact of postponements and buffer.

6.1. Solution Quality

First, we compare the average reduction in total delay of the policies ACA, Buffer, Postpone, and Deterministic compared with the base policy Fastest. The reduction is defined as the difference in value of the total delay from Fastest and value of the particular policy divided by the total delay from Fastest:

$$\frac{\text{Fastest} - \text{Policy}}{\text{Fastest}}.$$

We calculate the average reduction for the 2,000 runs per instance setting and then calculate the average over these instance settings. Figure 3 presents the results.

We observe that all policies, on average, significantly outperform the base policy. ACA achieves the highest reduction in total delay with a reduction of more than 60% on average. Thus, adding both a parameterized buffer and postponement leads to the best improvement compared with the other policies. A comparison of the results for the policies Buffer and Postpone demonstrates that, on average, the use of the parameterized buffer alone has a greater impact

on the reduction than considering only postponements. We demonstrate in subsequent discussions that the value of adding the parameterized buffer and/or postponement depends on the instance settings' characteristics. It is also interesting to see the improvements when using Deterministic. These improvements show that even a policy with deterministic costs and no postponements can reduce delay compared with sending the fastest vehicle.

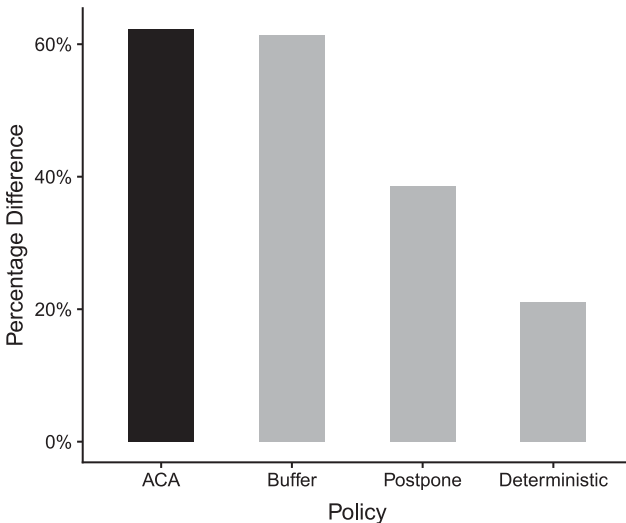
We now analyze the improvement from ACA for different classes of instance settings. We show the average improvement for different workloads and for the different COVs. The results are depicted in Figure 4. Figure 4(a) depicts the average improvement for different workloads, and each column represents the average over all of the COVs for each workload. Analogously, Figure 4(b) depicts the average improvement for varying COVs with each column representing the average for each COV over all of the workload values.

Generally, we observe improvements from 30% to more than 80%, demonstrating that the differences are highly dependent on workload and COV. We observe an increase in improvement with increasing workload. The greater the arrival rate, the more the resources become limited and bundling orders becomes necessary. In these cases, sending the fastest vehicle consumes too many resources to be flexible in the fulfillment of future orders.

We also observe a decrease in improvement with increasing randomness in the ready times. For increasing COVs, the buffer of the best ACA-tuning increases, hedging against the uncertainty in ready times. As the buffer increases, the ACA policy becomes more like the Fastest policy, and the improvement declines.

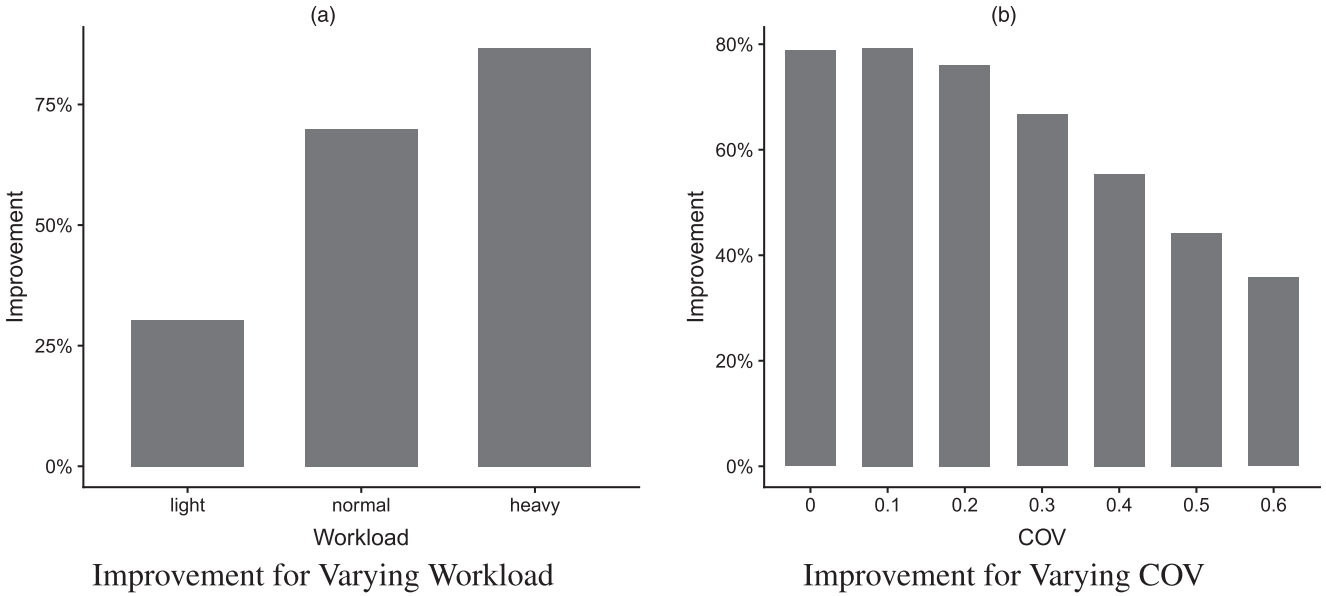
In the next two sections, we analyze the impact of postponement and buffer in the ACA and show when they are particularly important.

Figure 3. Overall Improvement in Total Delay as Compared with Policy Fastest



6.2. Postponement

In the following, we analyze the impact of postponing customers. We analyze the results for varying workloads and COVs. We also demonstrate the impact of increasing the number of postponements. To analyze the impact of postponement, we compare policy ACA to policy Buffer. The Buffer policy uses a buffer similar to ACA but does not postpone any orders. We calculate the improvement of ACA over Buffer similar to the equation in Section 6.1, but with Buffer as the base policy. We then analyze the improvement for varying work loads and COVs. Figure 5 presents the results. The figure follows the same structure as Figure 4. The left side shows the improvement for varying workloads. The right side shows the improvement for varying COVs.

Figure 4. Average Improvement of Policy ACA to Policy Fastest

Postponing customer orders enables information gain to improve assignments and the flexibility to bundle the postponed orders with potential new orders. We observe the value of postponement increasing with increasing workload. The explanation for this increase is twofold. First, the more customer orders, the more information that is gained for every minute and the higher the probability of finding a better assignment for the postponed customers. Second,

as the arrival rate increases, the likelihood for bundling opportunities increases.

We observe that, for a light workload, our postponement strategy provides slightly inferior results compared with not postponing. For these instance settings, there are very few new arrivals during the time that an order is postponed, and as a result, it is less likely that any new customer offers a valuable bundling opportunity for a postponed customer.

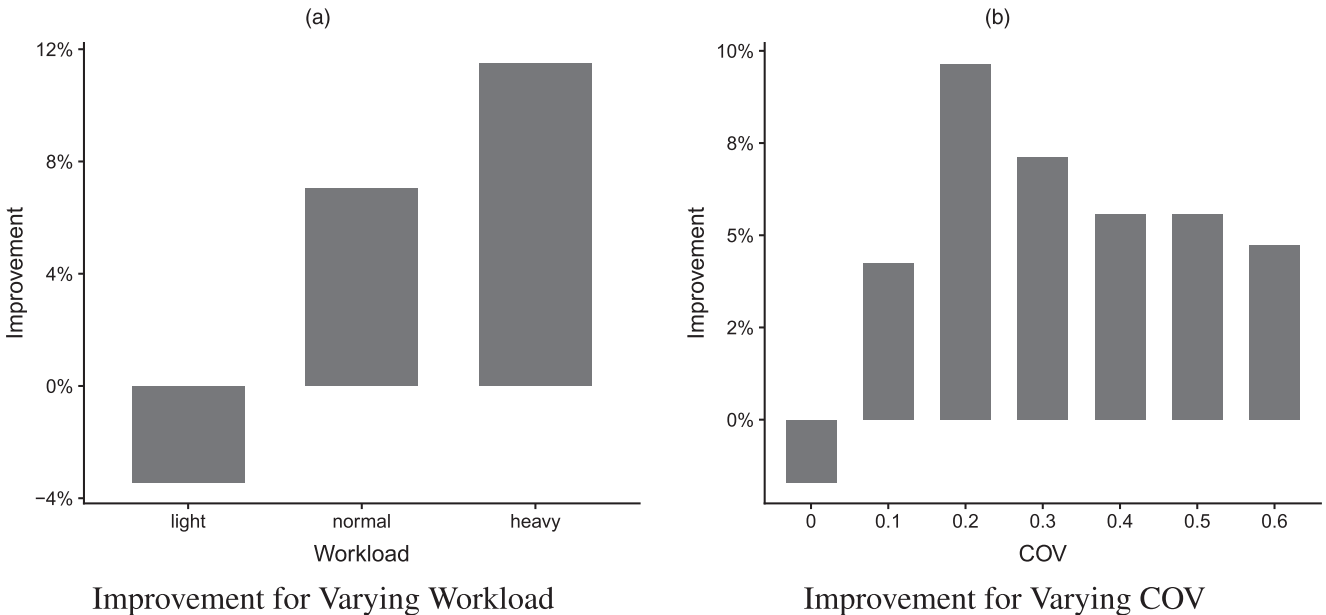
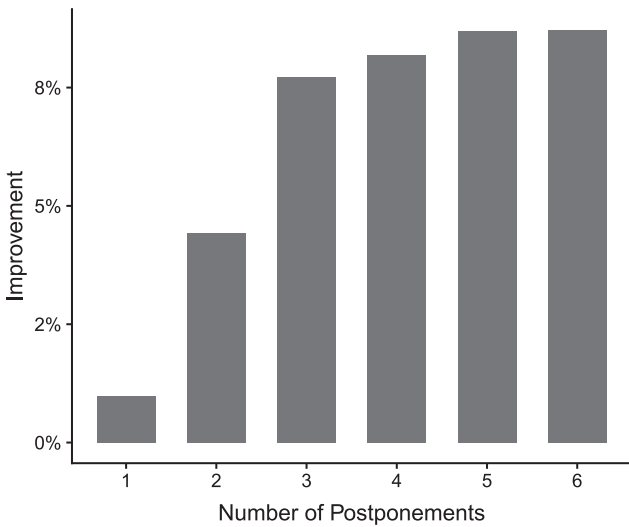
Figure 5. Average Improvement of Policy ACA to Policy Buffer

Figure 6. Average Improvement for Varying Number of Postponements



As the workload increases, there are more likely to be new arrivals that offer valuable bundling opportunities and, thus, superior performance.

The increasing value of postponement with increasing gain in information can also be observed for varying COVs. With more uncertainty in the ready times, the gain in information is high, at least up to a point. For very high COVs, the improvement slightly declines again. For these instances, even though the absolute difference in values increases, the absolute delay values are very high, leading to a relative decrease in improvement.

In our computational study, because of the number of simulations required for our comprehensive experiments, we postpone only two or three orders per decision point. In reality, it is possible to consider more orders for postponement. To this end, we next analyze the performance when postponing more than three orders. We focus on the seven instance settings with normal workload and COVs between 0.0 and 0.6. We systematically increase the maximum number of postponed orders from 0, 1, ..., 6. For all numbers of postponements, we follow the tuning procedure described in Section 5.3. Figure 6 shows the average improvement compared with policy Buffer. We observe that postponing one order adds only a small benefit. With two or more postponements, we observe significant improvements. We also observe that postponing more than three orders does not add substantial benefit compared with just postponing three orders.

6.3. Buffer

We now analyze the value of the buffer. We compare ACA to the policy Postpone for varying workloads and COVs. Figure 7, (a and b), presents the results.

In addition, the bottom two figures of Figure 7, (c and d), show the best average tuning values for the buffer for the workloads and COVs, respectively. Regardless of workload or COV, adding the buffer provides superior results. However, we also observe varying benefits of a buffer for different workloads and COVs. With increasing workload, the impact of the buffer decreases. The buffer is balancing hedging against delay (large buffer) with the flexibility to serve future orders (small buffer). With a light workload, resources are not scarce, and flexibility is unnecessary. In this case, ACA uses a large buffer to hedge against uncertainty. With increasing workload, flexibility becomes more important, and the buffer values decrease.

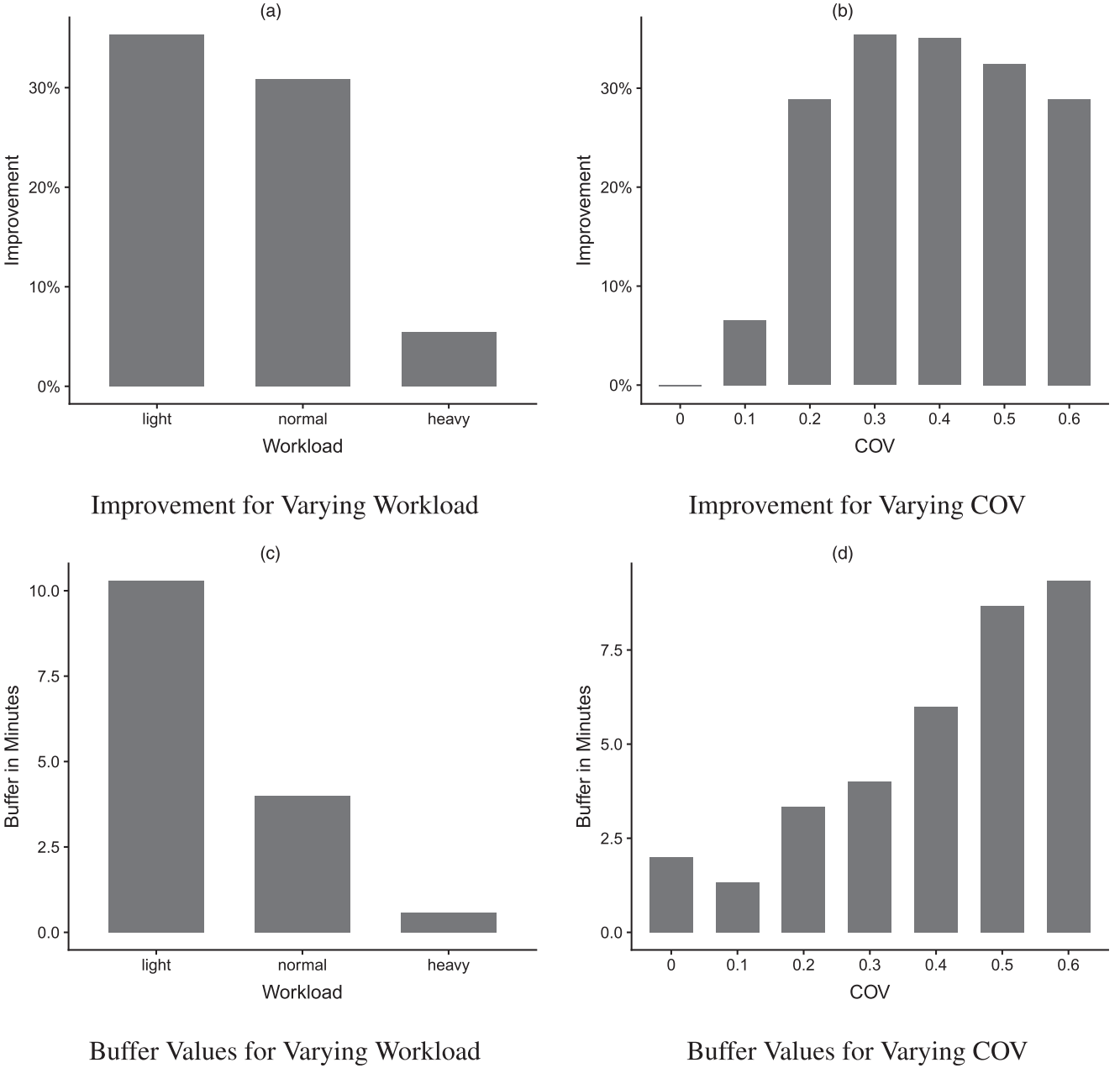
As shown in Figure 7(b), the buffer component of ACA is particularly valuable with increasing COV. We observe a significant increase in improvement with respect to increasing COV. With increasing COV, hedging against delay becomes important, and the buffer values increase as well. Similar to the results for postponement, we observe a stagnation and slight decrease with very high COVs because the absolute delay values for the corresponding instance settings are very high.

6.4. Variation in the Problem Dimensions

In our computational study, we assume that the ready times for all restaurants follow the same distribution and drivers bundle orders. To test the impact of these assumptions, we conduct additional experiments to analyze what happens if these assumptions do not hold: when the restaurants have heterogeneous ready-time distributions and the drivers are not bundling orders. In this section, we also consider the ability of the proposed approach to scale to meet the growing demand for restaurant meal delivery. To maintain the readability of the paper, the detailed results for the experiments are in the online appendix. In the following, we give a short summary.

6.4.1. Heterogeneous Ready Times and Customer-Dependent Buffers.

In the additional experiments, we group restaurants into three classes based on the time required to prepare the food, fast (five minutes expected ready time), fast casual (10 minutes), and gourmet (15 minutes). We follow the tuning procedure of ACA. The detailed results can be found in Online Appendix A.2. We observe that the improvements of our policy are similar to those in the main study. However, we also observe that the absolute delay values of all policies are higher if ready times are heterogeneous. This is not surprising as heterogeneous ready times mean that the problem has more variation. Particularly, because we characterize our instances by COV, a consequence of increasing the mean

Figure 7. Average Improvement of Policy ACA to Policy Postpone

ready time associated with a restaurant is that the restaurant's variance in ready times also increases.

The question becomes whether we need to design buffers to account for the increased variation, in particular, the variation associated with the different ready-time distributions. With this in mind, we adapt ACA to account for the heterogeneity in ready-time distributions. Instead of one global buffer, we use customer-dependent buffers. Notably, we increase the buffer for customers that are scheduled after a stop at a restaurant with a relatively high variance in ready times. Detailed results can be found in Online

Appendix A.3. Surprisingly, the customer-dependent policy performs similarly to ACA with a global buffer. This result indicates the complexity of determining customer-dependent buffers and offers an opportunity for future work.

6.4.2. Driver Routing. In our experiments, we assume that drivers bundle orders. That is, we assume that a driver can transport multiple orders at one time. The advantage of bundling is that the driver can choose to serve the restaurants one after the other before delivering the meals to customers. This practice has

intuitive appeal particularly when considering that restaurants are often clustered as they are in the Iowa City area (see Figure 2).

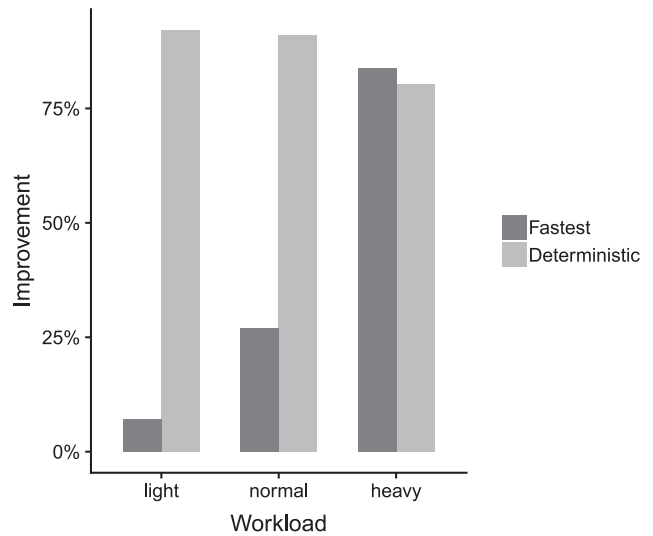
In additional experiments, we assume that drivers do not bundle using insertion, but rather always serve restaurant–customer pairs one after the other. Detailed results can be found in Online Appendix A.1. The results show that, even when not bundling, ACA performs significantly better than the Fastest policy. However, we observe an increase in the overall delay for all policies. The increase results from the fact that restaurants tend to be clustered in business zones such as downtown areas where bundling can be quite valuable.

6.4.3. Scalability. Finally, we analyze the ability of the algorithm to scale. In our computational evaluation, we focus on instance settings with 15 drivers, which is representative of the current order data in Iowa City. However, for larger systems and growing businesses, the number of orders and vehicles may be significantly higher. Thus, we analyze how ACA performs with an increasing number of orders. We focus on the ACA policy without postponement, which we have also called the Buffer policy. We focus on Buffer because postponements lead to much higher computation times when running the significant number of simulations necessary for both tuning and evaluation. We note though that, even at the scale of the experiments discussed in this section, each individual decision using ACA is fast enough for use in real time.

We generate instances with 30, 60, and 120 vehicles. For each fleet size, we test light workload (1.5 orders per vehicle per hour), normal workload (2.0), and heavy workload (2.5). We test instance settings with COV of 0.0, 0.1, ..., 0.6. This yields 63 instance settings with up to 2,400 orders per day. For each instance setting, we generate 2,000 training and 2,000 evaluation instances. We compare Buffer to the Fastest and Deterministic policies because they reflect either fast delivery or full exploitation of deadlines for future flexibility. Thus, they also give us an indication about the relative resource availability of the instances.

We calculate the average improvement of Buffer compared with Fastest and Deterministic. We then average the results with respect to workload and number of vehicles. Figure 8 presents the results for the varying workloads. The x -axis depicts the workload, and the y -axis shows the improvement of Buffer compared with Fastest and Deterministic. We observe that Buffer outperforms the other two policies. The improvement with respect to Deterministic remains nearly the same regardless of the workload. Thus, hedging against random ready times is always beneficial. The improvement compared with Fastest depends

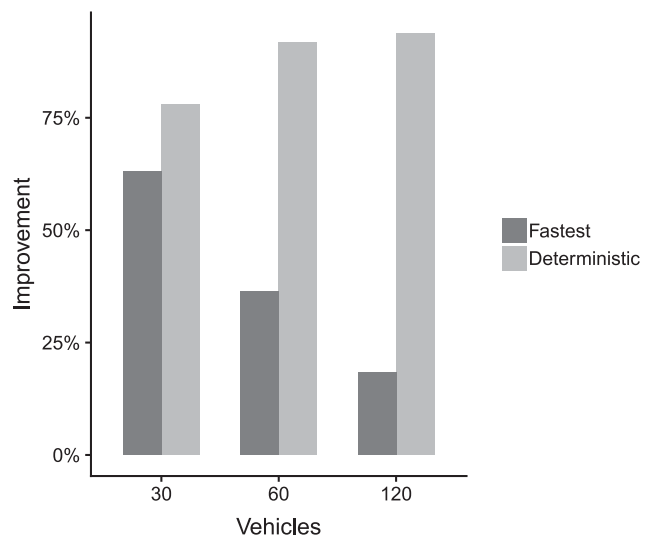
Figure 8. Improvement of Buffer for Large-Scale Instances and Varying Workload



on the workload. For a light workload, policy Fastest achieves nearly similar results. With increasing workload, the improvement of Buffer becomes substantial. These results are analogous to those of Section 6.1; sending the fastest vehicle is a reasonable strategy when sufficient resources are available.

Finally, we analyze the performance with an increasing numbers of vehicles. Figure 9 shows the results. Again, the improvement of Buffer compared with Deterministic is always high. However, with increasing numbers of vehicles, the improvement compared with Fastest decreases to about 20%. This decrease is the result of the fact that, with more vehicles, there is a greater likelihood of having a vehicle close to the restaurant associated with each order. This reduces the average travel time, increasing efficiency.

Figure 9. Improvement of Buffer for Increasing Fleet Size



7. Managerial Analysis

In this section, we analyze how our policy impacts the objectives for the four stakeholders involved, customers, restaurants, drivers, and the meal delivery company.

7.1. Measures

As mentioned in the introduction, for a meal delivery company to be successful and grow its own business, it must satisfy the needs of the customers, restaurants, and drivers. Thus, it is important to assess how different policies impact each of these stakeholders. We identify measures relevant to customers, restaurants, and drivers, respectively.

7.1.1. Customers: Delay. Customers want reliable and fast service. This desire is reflected in the use of the expected total delay as in our MDP objective. However, long delays can create customer dissatisfaction and can impact retention of these customers. Thus, we also compute the average maximum amount of delay experienced by a customer in the course of a day/simulation run. To better understand how the policies impact the solutions, we also compute the average number of deliveries that occur within different delivery time ranges. We visually describe delay and the next presented measure “freshness” in Figure 10.

7.1.2. Restaurants: Freshness. To satisfy their customers and grow their businesses, restaurants need their food to be fresh when it arrives to the customers. For this purpose, we define two measures: the average freshness of the food and the average maximum unfreshness. The average freshness is the average time between ready time of an order at a restaurant and delivery time to the customer. The average maximum unfreshness is computed based on the maximum time between ready time and delivery time in the course of a day/simulation run.

7.1.3. Drivers: Number of Orders. A driver’s objective is to make as many deliveries as possible to maximize wages. Because the orders are not always divided

evenly among the drivers, we define measures capturing fairness. The first measures are the average minimum and maximum numbers of services per day/simulation run. To understand what happens over a longer term, we introduce measures considering the drivers for a month (20 days) and then report the average minimum and maximum numbers of orders over the drivers for the 20 days. To compare our policies, we also compare the span between the driver with the maximum number and the driver with the minimum number of orders on a daily basis and over 20 days. These measures allow us to understand how evenly the orders are distributed.

7.2. Customers

In our main objective, we already focus on the customers with the sum of delay or the average delay per customer. However, customers may tolerate short delay but may be unhappy with long delays. Thus, in this section, we analyze the daily average maximum delay a customer experiences. Further, customers are not necessarily pleased with unexpectedly early deliveries (Zeithaml, Parasuraman, and Berry 1990). As a result, we also analyze how ACA distributes the delivery times compared with the current practice policy.

7.2.1. Maximum Delay. To analyze the policies’ impact on long delays, we calculate the average maximum delay experienced by a customer per day. Figure 11 shows the improvement of ACA and benchmarks with the respect to the policy Fastest across all instance settings. The axes are analogous to those in Figure 3 as are the results. Policies ACA and Buffer reduce the maximum delay as well as the average delay, and again, Postpone and Deterministic create improvements but not as much as with a parameterized buffer. In fact, even for the “hardest” instance setting with 300 orders and COV of 0.6, the average maximum delay a customer experiences per day is not more than 15 minutes for ACA, but is more than 24 minutes for policy Fastest. These results indicate that

Figure 10. Measures

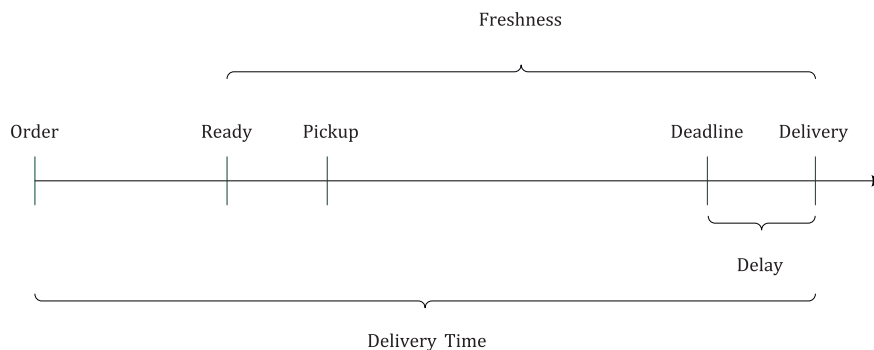
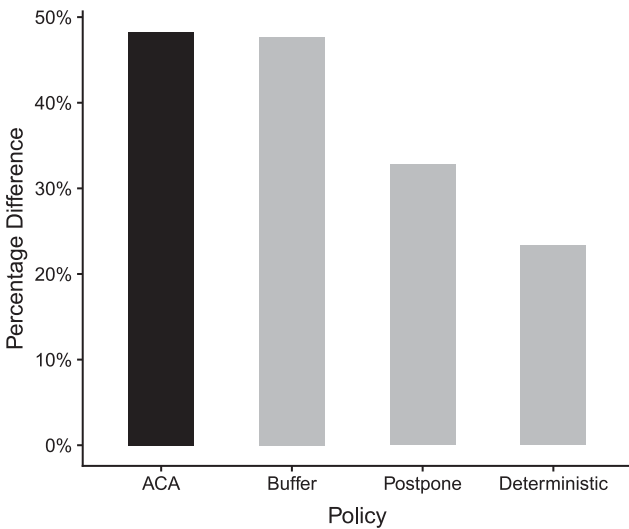


Figure 11. Overall Reduction in Average Maximum Delay Compared with Policy Fastest



the reduction in the average total delay for ACA is not obtained by sacrificing service quality for a minority of customers. Overall, ACA improves customer experience on average and also in the worst case.

7.2.2. Reliability in Delivery Times. We complete our analysis related to customer service by analyzing how delivery times of ACA differ from current practice policy Fastest that seeks to minimize the overall delivery time. As illustrated in Figure 10, delivery time is the time from when a request is made until when the meal arrives at the customer. As discussed, long delivery

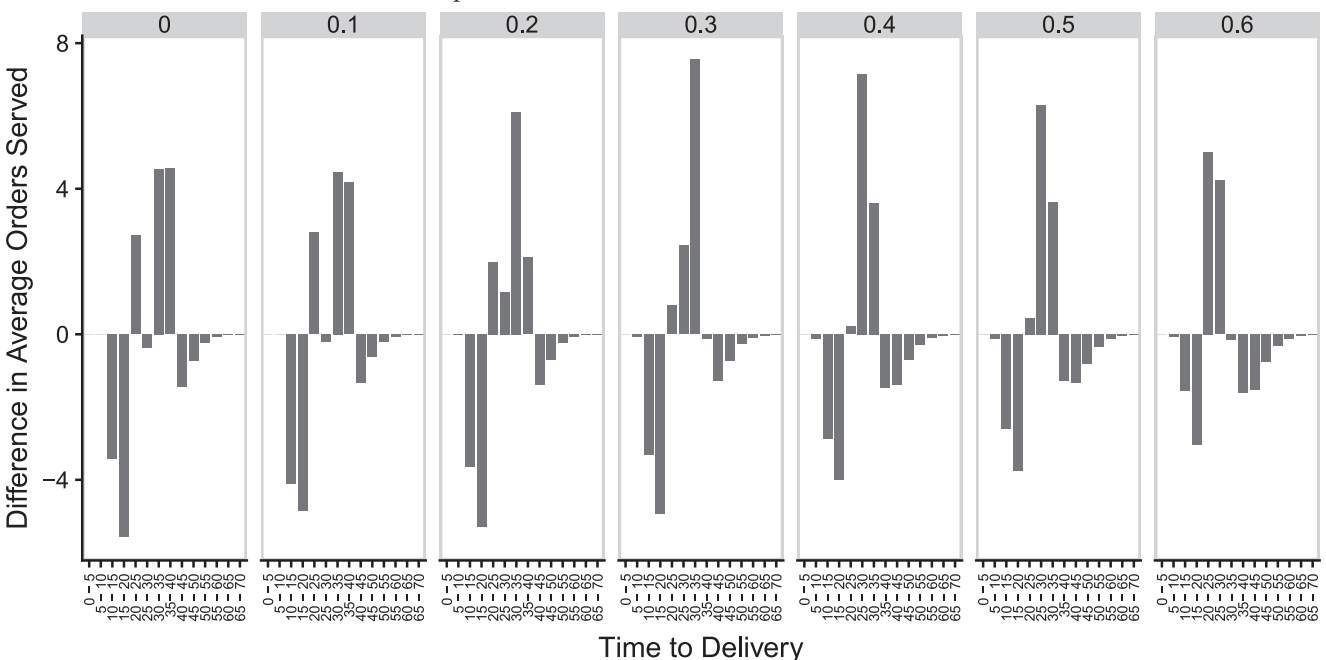
times cause delay and customer dissatisfaction. However, unexpectedly early delivery does not benefit the objective and may even lead to dissatisfaction because customers are not ready to receive the order.

To compare the delivery time distribution for each policy, we record the number of orders that have delivery times between 0–5 minutes, 5–10 minutes, ..., 65–70 minutes for each simulation run and average them for each instance setting. We then compute the difference in these values (average from ACA – average from Fastest) between the two policies. Thus, in case of a positive value, ACA conducts more services with these delivery times than Fastest.

We show the results for the instances with normal workload of 240 expected orders. However, the results for light and heavy workload show the same pattern. Figure 12 presents the differences by COV. The x -axis indicates the delivery time ranges, and the y -axis depicts the average number of differences in services between the two policies in each range. Comparing the two policies, we observe that ACA serves fewer customers fast (under 20 minutes) but also fewer customers late (more than 40 minutes). Thus, it concentrates delivery times in the area of 25 to 40 minutes. The effect of ACA is to reduce the number of fast deliveries while also reducing the number of late deliveries. This is a trade-off likely to satisfy customers.

With increasing COV, we also observe a shift to the left in the largest difference between ACA and Fastest. A comparison with Figure 7 shows that the concentration of deliveries in these time intervals is reflective of the best time buffer for each respective COV.

Figure 12. Normal Workload with 240 Expected Orders

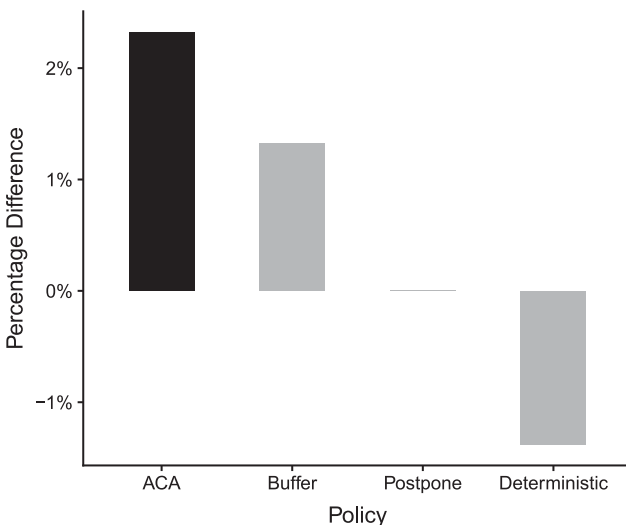


7.3. Restaurants

We now analyze how our proposed policy impacts the performance for the restaurants. Restaurants seek good reviews and returning customers. In addition to the timely arrival of the food or the delay, this outcome depends on the state of the food when it arrives at a customer. Fresh food increases the probability of future orders for the restaurant (and delivery company), and cold food may lead to bad reviews. Given that the objective of the optimization problem in this paper is delay, it is possible that the proposed policy satisfies the delivery deadline but does so by letting food sit after it is ready. Thus, we analyze how our policy impacts the average freshness of the food.

Figure 13 shows the improvement of the policies in average freshness across all instances relative to the policy Fastest. Both ACA and Buffer improve upon the base policy. Yet, compared with the delay measure, the improvements of the policies relative to Fastest are much smaller in this case. The improvement of policy ACA is only 2.3%. The performance in terms of freshness of policy Postpone is indistinguishable from Fastest, and the average freshness decreases with policy Deterministic. The reason for the reduced performance gains and even loss is that the freshness is not surprisingly strongly connected to the delivery time, and Fastest myopically minimizes delivery time. Thus, the base policy Fastest does relatively well, at least on average, in terms of delivering fresh food. However, as already discussed, ACA avoids very long delivery times and, therefore, extremely “unfresh” food. We present the analysis of the policies’ impact on maximum unfreshness in Online Appendix A.4.

Figure 13. Improvement in Freshness as Compared with Policy Fastest



7.4. Drivers

In this section, we analyze how the policies impact the drivers. Drivers’ earnings partially depend on the number of deliveries performed. Thus, we analyze the change in assignment of the orders to the drivers in the short and medium term.

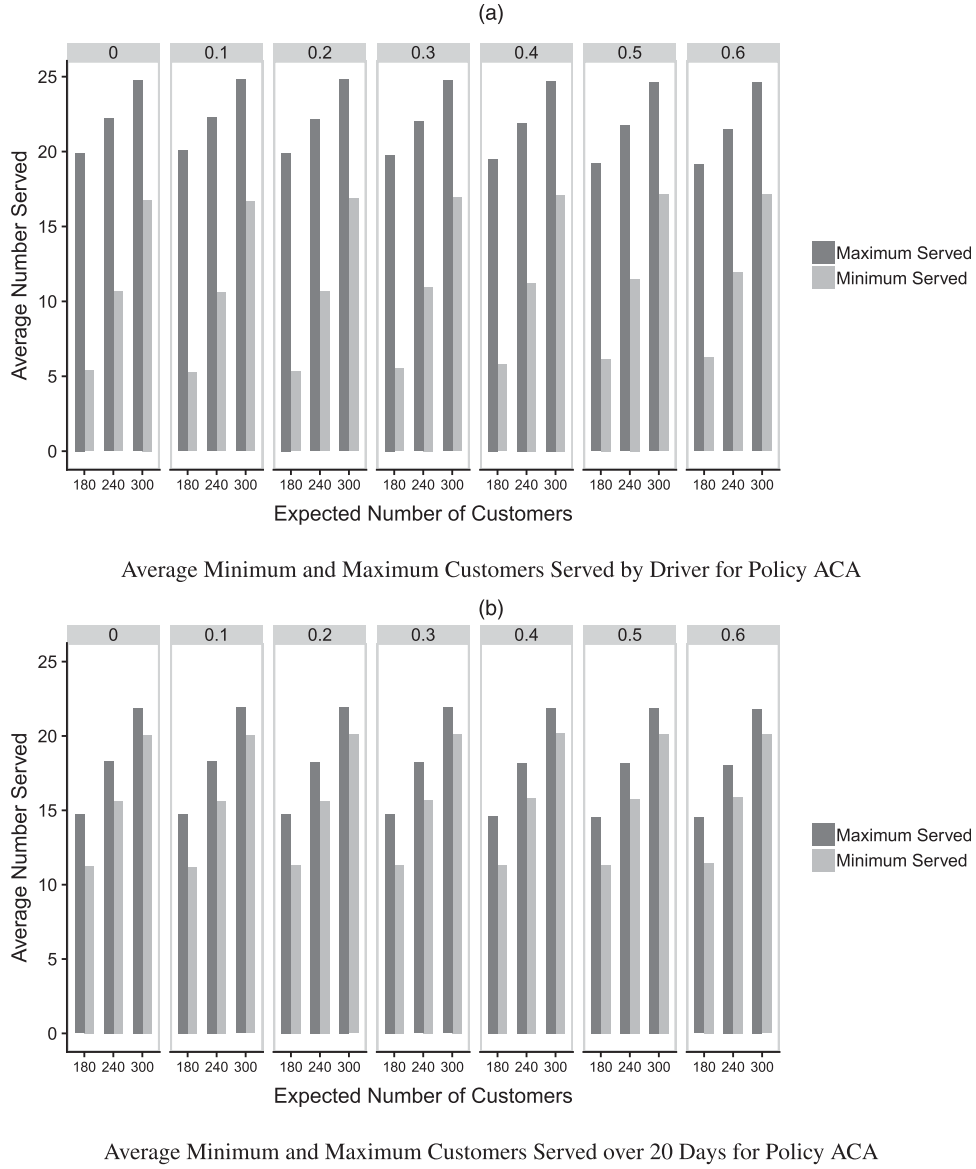
Our policy ACA generates flexibility by bundling orders (see Online Appendix A.1 for a detailed analysis on bundling). Bundling operations impact driver assignments. It may be that some drivers may conduct significantly more deliveries than others, leading to imbalanced earnings. In this section, we analyze how ACA changes the number of assignments to drivers as measured by the maximum number of customers served by any driver and the minimum number of customers served by any driver. If the disparity is sufficiently large, drivers interpret the assignment policies as favoring some drivers over others, impacting driver retention.

Figure 14(a) shows the average minimum and maximum number of orders served per day by drivers when policy ACA is used. The figure breaks out the results by expected number of customer orders and COV. The figure shows that the difference in customers served between the driver serving the maximum and the driver serving the minimum can be significant with the greatest disparities arising when the expected orders are lower. The reason for these disparities is that, when a driver is near a cluster of busy restaurants, it becomes advantageous to bundle and assign that driver to additional orders in that area.

Figure 14(b) presents analogous results to those in Figure 14(a), but first averages each driver’s number of customers served over 20 days or approximately one month of workdays. To compute the 20-day values, we use our existing runs and batch them into 20-day groups. For each set of 20 out of our 2,000 runs, we summarize the services for each driver and find the minimum and maximum number of services among the drivers over that 20-day period. We then take the average over the $\frac{2,000}{20} = 100$ values. In comparison with the results shown in Figure 14(a), Figure 14(b) shows that, over time, the disparities between the orders served by the driver serving the maximum and the driver serving the minimum greatly decreases. This decrease in disparity results because the driver who benefits from the additional orders provided by bundling is random on each day. Thus, the differences average out over time. Nonetheless, to retain drivers, it is important for managers to educate drivers on the daily fluctuations.

Online Appendix A.5 presents comparison of ACA to Fastest in terms of fairness. The results show that differences between the two policies are minimal

Figure 14. Examination of Maximum and Minimum Customers Served for Policy ACA



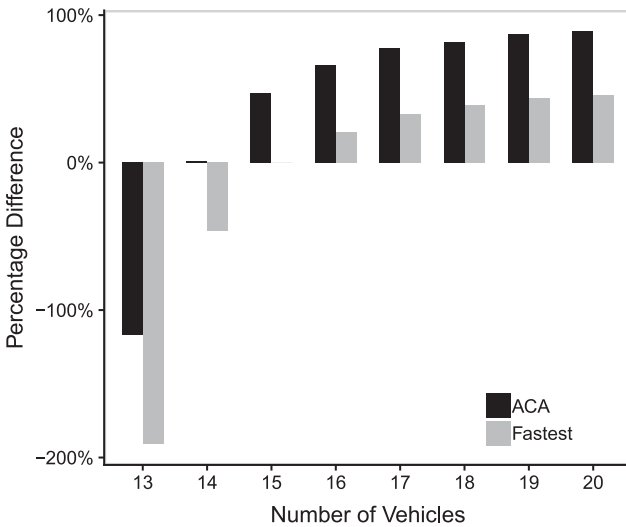
when considered over a 20-day period. Further, we show in the next section that ACA fosters business growth while keeping the number of drivers fixed. These outcomes offer increasing earning opportunities for each driver.

7.5. Meal Delivery Company

The meal delivery company seeks to satisfy the demands of customers, restaurants, and drivers. As we have shown in the previous section, ACA meets this challenge. Yet the company seeks to satisfy its stakeholders as a means of fueling its own growth. Importantly, growth cannot come at the expense of its stakeholders. The proposed ACA policy offers this opportunity. In Section 6.4, we show that the ACA policy can handle the larger numbers of vehicles and customers that

come with growth. Here, we demonstrate both that ACA can provide a higher level of customer service using fewer vehicles than Fastest and that, with the same fleet size, ACA can serve more customers than Fastest without sacrificing service quality.

We first show that ACA can provide a higher level of customer service using fewer vehicles than Fastest. To demonstrate our point, we focus on the instance setting with normal workload and COV of 0.2. We vary the number of drivers between 12 and 20 and analyze how the solution quality changes for policies ACA and Fastest. We benchmark the solution quality with the setting of 15 vehicles, normal workload, and COV of 0.2 using policy Fastest. The results are shown in Figure 15. The x -axis shows the number of vehicles. The y -axis shows the improvement with

Figure 15. Vehicles Needed by Policies ACA and Fastest to Provide Comparable Solution Quality

respect to 15 vehicles, normal workload, and COV of 0.2 using policy Fastest. As expected, we observe a general increase in improvement with respect to the number of vehicles for both policies. Yet, even with 20 vehicles, policy Fastest is not able to achieve the solution quality of policy ACA and 15 vehicles. For policy ACA, even with only 14 vehicles, the solution quality still outperforms Fastest with 15 vehicles. These observations are important because, by providing the same solution quality with fewer drivers, ACA allows a company to provide more opportunities for drivers without affecting customers.

To show that the proposed policy ACA can serve more customers than policy Fastest while also providing better customer service, we compare the instance settings with COV of 0.2 with both normal (240 expected customers) and heavy workload (300). For the instance setting with a heavy workload, we apply policy ACA, and for the normal workload, we apply Fastest. In this case, because the heavy workload instance requires the service of more customers, the total delay is 10.0% higher than in the normal workload case. Yet, having been applied to the heavy workload case, ACA is serving 60 more customers on average than Fastest, and the average delay *per customer* decreases. Thus, the ACA enables business growth while providing better customer service and offering drivers more opportunities.

8. Conclusion

In this paper, we introduce the restaurant meal delivery problem. We propose an ACA policy. ACA relies on a time buffer and postponement to account for the dynamism and uncertainty in the problem. The results of the computational study demonstrate that the proposed policy outperforms intuitive benchmarks

and a benchmark meant to mimic current practice by many companies. We also show that the policy is capable of scaling to large number of vehicles and customers.

The RMDP involves many stakeholders who have different objectives and who must be satisfied if restaurant meal delivery companies are to build successful businesses. We demonstrate the policy improves the objectives of all stakeholders, particularly in relation to a policy that assigns the closest driver, a policy used in practice by some meal delivery companies. ACA offers the opportunity for growth while not diminishing customer satisfaction and while allowing drivers to increase their wages.

With its rapid growth and dynamic market, restaurant meal delivery offers a promising area of research for transportation science and logistics researchers. Future research can consider extensions of the proposed model that account for driver repositioning strategies and even drivers' rejections of orders. Further, model extensions could account for shift scheduling. In addition, some restaurant meal delivery companies are beginning to use restaurant- or the combination of restaurant and customer-dependent delivery deadlines. We do not consider this feature in our model though our model can be easily extended to include restaurant- and customer-dependent deadlines.

Methodologically, our policy currently uses a static time buffer. Although the initial experiments on customer-dependent buffers present in this work show no significant improvement, future research could seek to identify more effective strategies for customer- or state-dependent buffers. Importantly, the proposed customer-dependent approach relies on the variance of the ready times at the restaurant associated with an order. It might be valuable to consider as well where an order falls in a sequence of visits. Doing so would require learning the distributions of customer delivery sequences as these cannot be derived analytically. Alternatively, one might consider developing a value function approximation approach.

From the business standpoint, there are a number of features of the business model to be explored. Most notably, there is no research on the value of incentive schemes for both customers and restaurants. For instance, there could be discounted delivery fees for customers who are willing to take delivery a half hour later than the regular deadline. On a related note, restaurants that guarantee ready times could receive discounts on their delivery fees. Determining when, which, and what size incentives are unexplored questions. Another interesting avenue could be to negotiate assignment decisions, especially in combination with driver rejections. Further, as consolidation continues among restaurant delivery companies, restaurants have begun to express concerns

about the rising fees associated with the service. Future work could explore what fees delivery companies should charge and what level of fees restaurants should accept.

Acknowledgments

The authors thank the editors and the referees for their very valuable support and constructive suggestions. They also thank Jon Sewell from CHOMP and Brian Wadsworth from OrderUp for their valuable insights into the restaurant meal delivery business.

References

- Archetti C, Jabali O, Speranza MG (2015) Multi-period vehicle routing problem with due dates. *Comput. Oper. Res.* 61:122–134.
- Archetti C, Feillet D, Mor A, Speranza MG (2018) An iterated local search for the traveling salesman problem with release dates and completion time minimization. *Comput. Oper. Res.* 98:24–37.
- Arda Y, Crama Y, Kronus D, Pironet T, Van Hentenryck P (2014) Multi-period vehicle loading with stochastic release dates. *EURO J. Transportation Logist.* 3(2):93–119.
- Azi N, Gendreau M, Potvin J-Y (2012) A dynamic vehicle routing problem with multiple delivery routes. *Ann. Oper. Res.* 199(1):103–112.
- Berbeglia G, Cordeau J-F, Laporte G (2010) Dynamic pickup and delivery problems. *Eur. J. Oper. Res.* 202(1):8–15.
- Boscoe FP, Henry KA, Zdeb MS (2012) Nationwide comparison of driving distance vs. straight-line distance to hospitals. *Professional Geographer: J. Assoc. Amer. Geographers* 64(2):188–196.
- Cattaruzza D, Absi N, Feillet D (2016) The multi-trip vehicle routing problem with time windows and release dates. *Transportation Sci.* 50(2):676–693.
- Cheng A (2018) Millennials are ordering more food delivery, but are they killing the kitchen, too? *Forbes Online* (June 26), <https://www.forbes.com/sites/andriacheng/2018/06/26/millennials-are-ordering-food-for-delivery-more-but-are-they-killing-the-kitchen-too/#c598008393e1>.
- Cortés CE, Núñez A, Sáez D (2008) Hybrid adaptive predictive control for a dynamic pickup and delivery problem including traffic congestion. *Internat. J. Adaptive Control Signal Processing* 22(2):103–123.
- Cortés CE, Sáez D, Núñez A, Muñoz Carpintero D (2009) Hybrid adaptive predictive control for a dynamic pickup and delivery problem. *Transportation Sci.* 43(1):27–42.
- Fagerholt K, Foss BA, Horgren OJ (2009) A decision support model for establishing an air taxi service: A case study. *J. Oper. Res. Soc.* 60(9):1173–1182.
- Fu MC (2015) *Handbook of Simulation Optimization*, vol. 216 (Springer).
- Ghiani G, Manni E, Romano A (2018) Scalable anticipatory policies for the dynamic and stochastic pickup and delivery problem. *EURO J. Transportation Logist.*, ePub ahead of print April 19, <https://doi.org/10.1007/s13676-018-0124-0>.
- Ghiani G, Manni E, Quaranta A, Triki C (2009) Anticipatory algorithms for same-day courier dispatching. *Transportation Res. Part E Logist. Transportation Rev.* 45(1):96–106.
- Hyttiä A, Penttinen A, Sulonen R (2012) Non-myopic vehicle and route selection in dynamic DARP with travel time and workload objectives. *Comput. Oper. Res.* 39(12):3021–3030.
- Isaac M (2016) The downside of delivery: Delivery start-ups face road bumps in quest to capture untapped market. *New York Times* (February 11), B1.
- Klapp MA, Erera AL, Toriello A (2016) The one-dimensional dynamic dispatch waves problem. *Transportation Sci.* 52(2):402–415.
- Klapp MA, Erera AL, Toriello A (2018) The dynamic dispatch waves problem for same-day delivery. *Eur. J. Oper. Res.* 271(2):519–534.
- Macmillan D (2016) Uber prepares to launch meal-delivery service in 10 cities. *Wall Street Journal* (January 21), B8.
- Maze J (2016) Not everything delivers: Saying no to delivery. Accessed July 12, 2016, <http://nrm.com/operations/not-everything-delivers-saying-no-delivery/>.
- Mitrović-Minić S, Laporte G (2004) Waiting strategies for the dynamic pickup and delivery problem with time windows. *Transportation Res. Part B: Methodological* 38(7):635–655.
- Mitrović-Minić S, Krishnamurti R, Laporte G (2004) Double-horizon based heuristics for the dynamic pickup and delivery problem with time windows. *Transportation Res. Part B: Methodological* 38(8):669–685.
- Muñoz Carpintero D, Sáez D, Cortés CE, Núñez A (2015) A methodology based on evolutionary algorithms to solve a dynamic pickup and delivery problem under a hybrid predictive control approach. *Transportation Sci.* 49(2):239–253.
- Núñez A, Cortés CE, Sáez D, De Schutter B, Gendreau M (2014) Multiobjective model predictive control for dynamic pickup and delivery problems. *Control Engrg. Practice* 32:73–86.
- Powell WB (2011) *Approximate Dynamic Programming: Solving the Curses of Dimensionality*, 2nd ed., Wiley Series in Probability and Statistics (John Wiley & Sons, Inc., Hoboken, NJ).
- Powell WB (2019) A unified framework for stochastic optimization. *Eur. J. Oper. Res.* 275(3):795–821.
- Psaraftis HN, Wen M, Kontovas CA (2016) Dynamic vehicle routing problems: Three decades and counting. *Networks* 67(1):3–31.
- Reyes D, Erera AL, Savelsbergh MW (2018) Complexity of routing problems with release dates and deadlines. *Eur. J. Oper. Res.* 226(1):29–34.
- Reyes D, Erera AL, Savelsbergh MW, Sahasrabudhe S, O’Neil R (2018) The meal delivery routing problem. Accessed December 20, 2018, http://www.optimization-online.org/DB_FILE/2018/04/6571.pdf.
- Rosenkrantz DJ, Stearns RE, Lewis PM (1974). Approximate algorithms for the traveling salesperson problem. *IEEE Conf. Record 15th Annual Sympos. Switching Automata Theory* (IEEE) 33–42.
- Sáez D, Cortés CE, Núñez A (2008) Hybrid adaptive predictive control for the multi-vehicle dynamic pick-up and delivery problem based on genetic algorithms and fuzzy clustering. *Comput. Oper. Res.* 35(11):3412–3438.
- Sayarshad HR, Chow JY (2015) A scalable non-myopic dynamic dial-a-ride and pricing problem. *Transportation Res. Part B: Methodological* 81(2):539–554.
- Schilde M, Doerner KF, Hartl RF (2011) Metaheuristics for the dynamic stochastic dial-a-ride problem with expected return transports. *Comput. Oper. Res.* 38(12):1719–1730.
- Schilde M, Doerner KF, Hartl RF (2014) Integrating stochastic time-dependent travel speed in solution methods for the dynamic dial-a-ride problem. *Eur. J. Oper. Res.* 238(1):18–30.
- Shelbourne BC, Battarra M, Potts CN (2017) The vehicle routing problem with release and due dates. *INFORMS J. Comput.* 29(4):705–723.
- Statista Survey (2016) How often do you order food for takeout? Accessed December 6, 2016, <https://www.statista.com/statistics/319662/frequency-of-ordering-food-for-takeout-or-delivery-us/>.
- Ulmer MW (2020) Dynamic pricing and routing for same-day delivery. *Transportation Sci.* 54(4):1016–1033.
- Ulmer MW, Thomas BW (2018) Same-day delivery with heterogeneous fleets of drones and vehicles. *Networks* 72(4):475–505.
- Ulmer MW, Thomas BW, Mattfeld DC (2019) Preemptive depot returns for dynamic same-day delivery. *EURO J. Transportation Logist.* 8(4):327–361.
- Ulmer MW, Goodson JC, Mattfeld DC, Thomas BW (2017) Modeling dynamic vehicle routing problems: A literature review and framework. Accessed August 2, 2017, https://www.researchgate.net/publication/313421699_Route-Based_Markov_Decision_Processes_for_Dynamic_Vehicle_Routing_Problems.

- U.S. Census Bureau (2017) Annual estimates of the resident population: April 1, 2010 to July 1, 2016. Accessed September 28, 2017, <https://factfinder.census.gov/faces/tableservices/jsf/pages/productview.xhtml?src=bkmk>.
- Voccia SA, Campbell AM, Thomas BW (2017) The same-day delivery problem for online purchases. *Transportation Sci.* 53(1):167–184.
- Vonolfen S, Affenzeller M (2016) Distribution of waiting time for dynamic pickup and delivery problems. *Ann. Oper. Res.* 236(2):359–382.
- Zaleski O (2018) This meal delivery company was just another struggling startup until wooing restaurants paid off. *BusinessWeek Online* (December 19). <https://www.bloomberg.com/news/articles/2018-12-19/doordash-was-left-for-dead-until-wooded-restaurateurs-began-to-pay>.
- Zeithaml VA, Parasuraman A, Berry LL (1990) *Delivering Quality Service: Balancing Customer Perceptions and Expectations* (Simon and Schuster, New York).