The Restaurant Meal Delivery Problem: Dynamic Pick-Up and Delivery with Deadlines and Random Ready Times

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

October 12, 2017

Abstract

We consider a stochastic dynamic pickup and delivery problem. A fleet of drivers delivers food from a set of restaurant to spontaneously ordering customers. The objective is to dynamically control a fleet of drivers in a way that avoids delays with respect to customers deadlines. The sources of stochasticity for this problem are twofold: first, the customers are unknown until they place an order. Second, the time the food is ready at the restaurant is unknown. To address these challenges, we present a parametrizable cost function approximation (CFA). To account for the stochasticity in customer requests, the CFA postpones the assignment decisions for selected customers allowing more flexibility in the response to new requests. Further, the CFA introduces time buffers to account for the uncertainty in the ready times of the food. Based on real-world data, we show how our CFA is able to improve service significantly for all stakeholders compared to current practice.

Key words: Restaurant meal delivery, dynamic vehicle routing, stochastic requests, stochastic ready times, cost function approximation

1 Introduction

People like to eat at home, but do not always like to prepare it. In a recent survey, 68% of survey 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 a seen a surge in companies that deliver restaurant meals to customers. These companies include startups like Grubhub, OrderUp!, 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 is typically promised between 30 to 40 minutes after the customer's call for a delivery fee between \$4 and \$7 (Macmillan, 2016). The market for food delivery in the US is expected to grow 79% in the next five years to a \$76 billion market (Franck, 2017).

To successfully run their businesses, restaurant meal delivery companies must satisfy several key stakeholders who 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. This fee serves as compensation for offering the 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 fee. However, late deliveries or meals that are no longer fresh can make customers dissatisfied. In serving customers, Zeithaml et al. (1990, p. 27) write, "reliability is the most critical dimension, regardless of the service being studied." Further, when orders arrive late, not only are customers less likely to use the delivery service again, but 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 on service. 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). Adding drivers drives down the ratio of orders to drivers and leads to lower driver pay. When the number of deliveries per hour is too small, companies face increased costs due to the need to recruit and train new drivers as existing drivers leave the service or as the result of countermeasures to retain current drivers (e.g. downside guarantee compensation).

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. Once a request is placed, the order is immediately relayed to the provider who assigns it to a driver who will pickup and deliver the order. The assignments of orders to drivers need not be immediate, and it is possible to bundle orders such that a driver is assigned multiple outstanding orders at one time.

Before delivery of an order, the driver must pickup the order from the associated restaurant. However, the time to prepare a customer's food at each restaurant is random. While the delivery company knows a distribution on the time, the provider 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 her/his 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 delivery deadlines of orders over the service day. For example, consider a request placed at 5:00 PM with a deadline of 5:40 PM. An order arriving at anytime 5:40 PM or earlier would contribute nothing to the objective, while an order arriving at 5:45 PM would incur a penalty of 5.

Delay free delivery to customers 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. While restaurant locations are known and fixed, customer locations are often unknown in advance of a request. Similarly, the time at which a customer will place an order and from what provider they will order is unknown. Further, the time required to prepare the order is random.

Due to these multiple challenges, our conversations with a service provider reveal that many currently assign the new order to the driver who is able to conduct the delivery fastest based on current information. This policy is myopic and neither accounts for future orders nor random ready times. Furthermore, sending the closest driver may limit the opportunities to assign orders to vehicles that are already en route to a nearby restaurant, thus freeing resources for the future. We use the term bundling to refer to the practice of assigning multiple orders at a time to a vehicle. To account for the sources of randomness and to enable bundling operations, we develop a parametrizable cost function approximation (CFA) based on a route-based Markov decision process (MDP) model. A CFA integrates a state-dependent cost-term in the determination of a decision to account for potential future developments. Powell (2017) notes that CFAs are an "important class of policy that has been overlooked by the research community." CFAs are particularly suited to large-scale optimization problems because they are able to "easily handle very high-dimensional data." In our case, we parameterize a time buffer that we use when computing the delay penalty incurred by a decisions. Like buffers in inventory problems, the time buffer allows us to account for the uncertainty in future orders as well as in the ready times at the restaurants. To obtain flexibility in our decision making, we also apply a postponement strategy that delays the assignment of selected customers.

We analyze our method based on a restaurant meal delivery service located in Iowa City. Iowa City has features in common with many cities where meal delivery occurs or a single delivery zone within larger cities. Comparisons with the current practice approach show the proposed policy results in significant advantages with respect to all stakeholders objectives. We show:

- For customers, our policy significantly reduces the delay per customer as well as the number of deadline violations.

- For restaurants, our policy also avoids delivery of exceptionally un-fresh food regularly observed by the current policy used in practice. This leads to potentially more orders in the future. We further observe that the uncertainty in ready times has a major impact on the solution quality.

- For drivers, our policy maintains the fairness among drivers in that, over time, all drivers serve almost the same number of orders. We further show that, with our policy, the same fleet size is able to provide better solution quality than current practice even as the number of orders per driver increases. The policy further allows the meal delivery company to grow and add more trips and thus 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. The computational study is shown in Section 5. 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 closely associated 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 related literature. 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 entitled “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

Table 1: Literature Classification

	Paper	Deadlines	Timing	Objective	Postponement	Scalability
DPDP	Mitrović-Minić and Laporte (2004)	✓				✓
	Mitrović-Minić et al. (2004)	✓				✓
	Cortès et al. (2008)	✓	✓	✓		
	Sàez et al. (2008)	✓		✓		
	Cortès et al. (2009)	✓		✓		
	Fagerholt et al. (2009)	✓				✓
	Ghiani et al. (2009)	✓		✓		
	Ghiani et al. (2017)	✓		✓		✓
	Schilde et al. (2011)	✓	✓	✓		
	Hyytiä et al. (2012)	✓		✓		✓
	Núñez et al. (2014)	✓		✓		
	Muñoz Carpintero et al. (2015)	✓		✓		
	Sayarshad and Chow (2015)	✓		✓		✓
	Vonolfen and Affenzeller (2016)	✓				✓
SDDP	Azi et al. (2012)	✓			✓	
	Klapp et al. (2016)				✓	
	Ulmer et al. (2016)					
	Voccia et al. (2017)	✓			✓	
	Klapp et al. (2017)				✓	
	Ulmer (2017)	✓				
	RMDP, CFA	✓	✓	✓	✓	✓

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 or not 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 or not the paper allows the assignment of requests to a vehicle to be postponed to allow for potentially better bundling of orders. Finally, the column "Scalability" refers to whether or not the method in associated paper can scale to handle large number of customers and vehicles.

2.1 Dynamic Pickup and Delivery

The DPDP is the problem which occurs when goods must be transported from unique pickup locations to unique dropoff locations. When people are transported rather than goods, the DPDP is often referred to as the dial-a-ride problem (DARP). Psaraftis et al. (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 et al. (2010). In our review, we focus on only those problems labeled as dynamic and stochastic by Psaraftis et al. (2016). Like ours, these papers address the dynamic problem by incorporating probabilistic information into the solution approach.

While there has been a number of papers addressing the DPDP, the applications are varied, and there does not seem to be a common approach. Mitrović-Minić and Laporte (2004) proposed waiting strategies for the strategies 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 et al. (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 et al. (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 contrast to most of the waiting approaches, our approach determines the parameters of the CFA through offline simulation allowing us to account explicitly for probabilistic information. In addition, our approach is designed to address the decision of assigning requests to vehicles in light of the uncertain ready times. Waiting strategies cannot address this element of our problem.

Related to Mitrović-Minić and Laporte (2004) is Mitrović-Minić et al. (2004). Mitrović-Minić et al. (2004) not only incorporates waiting strategies but builds routes using a strategy that discounts the combined cost and time consumption of customers who will be served later in the horizon. Such a method is effective because it accounts for the fact that greater density generated from customers who will request service in the future will in fact reduce the actual marginal cost of serving these customers. The double-horizon method of Mitrović-Minić et al. (2004) can, like our approach, be categorized as a CFA. The approaches differ in that the approach of Mitrović-Minić et al. (2004) is designed to handle uncertainty due to future requests while ours incorporates both the uncertainty of future requests as well as the uncertainty in ready times.

Cortès 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, Cortès et al. (2009) lookahead using a set of scenarios of the future covering a limited horizon. Because of computational challenges, Cortès et al. (2009) are limited to a two-step lookahead horizon. In contrast, by using offline simulation, we are able to determine the parameters of the CFA used in this paper looking across the entire horizon. In addition, because our approximations take place offline, 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 Cortès et al. (2009), but still solve problems of only 11 vehicles. Because Cortès et al. (2009) address a dynamic DARP and thus are transporting passengers, as in this paper, as in this paper, their objective does address the concerns of the customers. Núñez et al. (2014) adapt the method of Cortès et al. (2009) to multiple objectives in this case considering the trade off between operator and user costs.

Cortès et al. (2008) extend the methodology of Cortès et al. (2009) to include stochastic travel times. Like the random ready times in this paper, stochastic travel times impact customer service, and Cortès et al. (2009) seek to minimize the cost of waiting and travel times to customer. However, the resulting methodology is limited in its scalability. Sàez et al. (2008) introduce a more sophisticated procedure than that presented in Cortès et al. (2009) for developing predictions of future demand. Like Cortès et al. (2008), Schilde et al. (2011) consider a DPDP with stochastic travel times. Instead of the genetic algorithm for solving the hybrid predictive control problem as Cortès et al. (2008) do, Schilde et al. (2011) propose a variable neighborhood search to solve the probabilistic routing problems. The method is again requires realtime computation raising questions about its ability to scale, particularly when immediate decisions are needed as they are in the RMDP.

Hyytiä et al. (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 similar to exact dynamic programming except that actions are selected via an approximate Bellman Equation rather than exactly. This approximation is done on the second term of the Bellman Equation, the cost-to-go. Our CFA 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 et al. (2012) considers each vehicle individually and not their future interactions. Our method of tuning the CFA parameter implicitly accounts for these interactions. Sayarshad and Chow (2015) extend Hyytiä et al. (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 deadline. 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. While effective, the proposed method requires online computation and does not scale like our proposed approach.

Ghiani et al. (2017) seek a scalable method for a variant of the DPDP in which customers of different priorities are served over the problem horizon. Ghiani et al. (2017) proposed a policy-function approximation approach (PFA). Their PFA is characterized by a policy with a tunable parameter. Ghiani et al. (2017) propose a policy that 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 offline simulation. Conceptually, the PFA proposed in Ghiani et al. (2017) is similar to the CFA proposed in this paper. Because of the differences between the two problems, however, the PFA proposed in Ghiani et al. (2017) cannot be applied to the RMDP.

2.2 Same-Day Delivery

A recent variant of the DPDP is the same-day delivery problem (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 also 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 et al. (2012), Voccia et al. (2017), and Ulmer (2017) consider fleets of vehicles and only Voccia et al. (2017) and Ulmer (2017) consider customer deadlines or time windows. Both Azi et al. (2012) and Voccia et al. (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 (2017) relies on a value-function approximation approach that incorporates a lookup table. While 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. Other SDDP literature includes Klapp et al. (2017), Klapp et al. (2016), and Ulmer et al. (2016), all of which focus on a single vehicle case.

2.3 Further Related Literature

The authors are not aware of any literature that considers both stochastic customers and ready times. While 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 problem studied in this paper, the periods at which the known items become available for delivery is unknown. In contrast to the 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 et al. (2015), Cattaruzza et al. (2016), and Reyes et al. (pear) study ready times in a deterministic context.

3 Model

In this section, we first present a formal description of the RMDP in a problem statement. We then give an example for a decision state and model the problem as a route-based Markov decision process. In the problem statement, we introduce the notion of a "route plan." This notion is then used throughout the paper in both the model and method.

3.1 Problem Statement

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

Each vehicle V_i begins service at an initial position $N^V_{\text{init}} \in \mathcal{N}$. As 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. This includes waiting at locations and repositioning after a delivery was completed. However, we assume that these routing and sequencing procedures are known to the dispatchers. Because dispatcher and drivers communicate via mobile phones, diversions are not permitted while a driver is driving.

Orders occur according to a known stochastic process $\mathcal{F}^D$. 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. 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 $\mathcal{F}^R$. 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 will be assigned to which vehicles. Assignments can be postponed but, once made, are permanent. The assignment decisions combined with the drivers' sequencing and routing lead to a set of planned routes $\Theta = (\theta_1, \ldots, \theta_h)$, one route for each vehicle. Because drivers are self-employed, the plans are determined by the drivers based on our assignment decisions. Thus, we know the plans, but are not able to alter them. In practice, drivers often use an insertion-type heuristic to make these routing decisions. We will do similarly 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 planned arrival time:

$$\theta = ((N_1^\theta, a_1^\theta), (N_2^\theta, a_2^\theta), \ldots, (N_o^\theta, a_o^\theta)).$$

The planned arrival times a reflect the dispatcher's assumptions about when a vehicle will arrive at a certain location. These planned times will 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 plan represents the next location a vehicle will visit and when it arrives at this location. Because the vehicle is en route and will not be 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 conducted the delivery. The time $\alpha_{\text{real}}^D \in T_+$ represents the 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 will 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 are indicated by the rectangles. The planned arrival time at restaurants and customers is represented by the hexagons. For the purpose of presentation, we assume a Manhattan-style grid with travel times of 2.5 minutes per segment. We ignore service times for the purposes of this illustration. The mean of the ready time distribution at the restaurants is 10 minutes, and the deadline is $\bar{t} = 40$ minutes. At the left side of Figure 1, a decision point and according state is shown. The current point of time is $t = 60$ minutes. Customer D_1 ordered from Restaurant 1 at time $t^{D_1} = 55$. Customer 2 ordered from Restaurant 2 at time $t^{D_2} = 60$. The deadlines 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

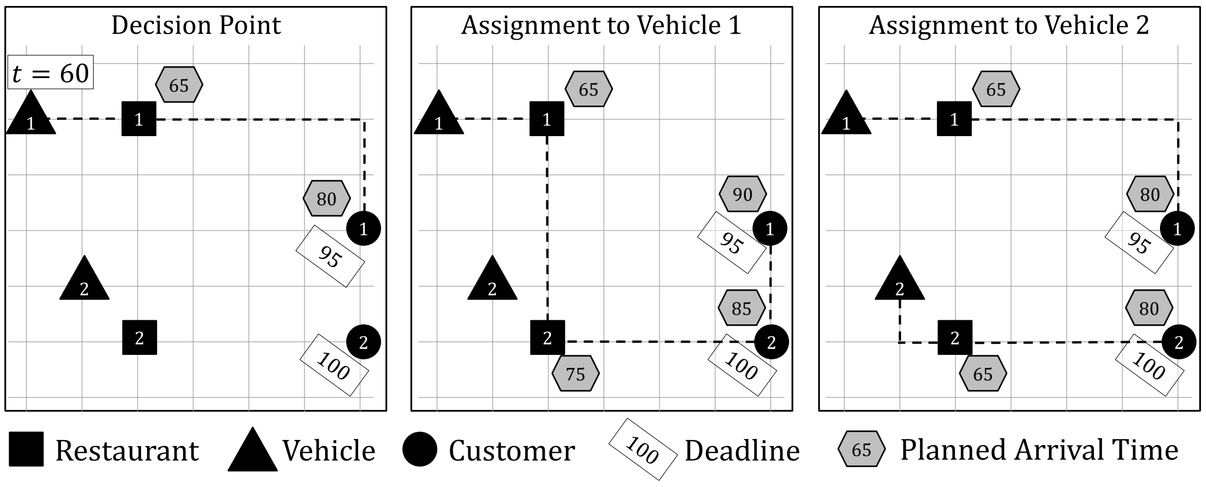


Figure 1: Example for the MDP Model

route 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. Decisions are now made about the assignment of Customer 2. This customer can be assigned to Vehicle 1, Vehicle 2, or the assignment can be postponed for an amount of time.

In the following, we describe the two potential assignments to Vehicle 1 or Vehicle 2. The resulting routing plans are depicted in the center and on the right side of Figure 1. The middle plan reflects the assignment of Customer 2 to Vehicle 1. The updated plan for Vehicle 1 is now

$$((R_1, 65), (R_2, 75), (D_1, 90), (D_1, 95)).$$

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. The arrival at Customer 2 is planned only at time 80 because the expected ready time for the food at Restaurant 2 is 10 minutes.

Once a decision is made, the vehicles proceed with their plan. Assume that at time $t = 105$, the next customer orders and that 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 $\alpha^{D_1}_{\text{real}} = 100$ at Customer 1 and $\alpha^{D_2}_{\text{real}} = 95$ at Customer 2. The realized delay at Customer 1 is 5 minutes. If Customer 2 is assigned to Vehicle 2, the realized arrival time is 90 at Customer 1 is 90 and 80 at Customer 2. All customers are served in time without delay.

3.3 Route-Based Markov Decision Process

In this section, we present a Markov decision process model (MDP) for the problem described previously. 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 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 due to the stochasticity in the ready times and the arrival of new orders.

In the following, we define the MDP components; decision point, decision state, decisions and stochastic transitions, both with associated costs.

Decision Points

Decision points are the points in time at which decisions are made during the horizon of the problem. In the RMDP, decision points $k = 0, \ldots, K$ occur in two cases. First, a decision point occurs when a new customer D_k 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, decision point k is then at time $t_k = t_{k-1} + \delta$. The initial decision point occurs when the first customer orders. The process terminates when all orders are served and the order horizon has ended, $t_K \geq t_{\text{max}}$.

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, the set of orders $\mathcal{D}_k$ including the new order D_k if applicable, and 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 request time of D, R^D the restaurant, V^D the assigned vehicle, and L^D the loading status, whether the order is already picked up at a restaurant. Parameter V^D is a number between 0 and $|\mathcal{V}|$ and indicates the assignment of order D. In the case that $V^D = 0$, 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 ($L^D = 0$).

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. A decision is feasible, if the following conditions hold:

1. The arrival time of the first entry of each route θ 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 drivers 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 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 times, the travel times, and the service times. 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. These costs may therefore be negative. For a given state S, to define C^d, we first define route plan $\widehat{\Theta}$, which may be the route plan Θ associated with the state S or the route plan Θ^x following a decision x at state S. We then introduce a delay-function Δ that, given a state S and a route plan $\widehat{\Theta}$, maps a route plan to the planned delay:

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

where a^D is the planned arrival time associated with customer D. The costs of decision x taken at state S_k are then calculated as

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

The costs are therefore the difference between updated planned delay $\Delta(S_k, \Theta_k^x)$ and currently planned delay $\Delta(S_k, \Theta_k)$.

Transition and Stochastic Costs

The stochastic transition is induced by the realization of exogenous information ω and transfers a state-decision pair to the next decision state $S_{k+1} = (S_k, x, \omega)$. The next decision state at time

t_{k+1} either occurs when a new customer orders or when the time for postponement δ has passed. Following the transition, the new routes Θ_{k+1} maintain the sequence of the routes Θ_k, but without all visits between t_k and t_{k+1}. Further, the arrival times are updated with respect to the observed ready times. In case an order for a customer D was picked up, the loading status of this customer is set to loaded ($L^D = 1$). Finally, in case a new customer request induced the transition at t_{k+1}, the new customer is added to $\mathcal{D}_{k+1}$.

The transition further results in stochastic costs C^s. These costs reflect the additional delay observed for the served customers compared to the planned delay. Let $\mathcal{D}_k^\omega$ be the number of served customers between t_k and t_{k+1}. For one of these served customers D, let a^D be the planned arrival time, t^D the order time, and α^D_{real} the realized arrival time. Then, the realized cost is defined as

$$C^{s,\omega}(S_k, \Theta_k^x, \omega) = \sum_{D \in \mathcal{D}_k^\omega} \left[\max(\alpha^D_{\text{real}} - (t^D + \bar{t}), 0) - \max(a^D - (t^D + \bar{t}), 0) \right]. \qquad (2)$$

Again, the costs can be negative.

The stochastic cost is defined as the expected cost over all realizations:

$$C^s(S_k, \Theta_k^x) = \sum_{\omega} P(\omega \mid S_k, \Theta_k^x) C^{s,\omega}(S_k, \Theta_k^x, \omega). \qquad (3)$$

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

$$\min_{\pi \in \Pi} \mathbb{E} \left[\sum_{k=0}^{K} C^d \left(S_k, \Theta_k, \Theta_k^{X^\pi(S_k)} \right) + C^s \left(S_k, \Theta_k^{X^\pi(S_k)} \right) | S_0 \right]. \qquad (4)$$

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 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 $a^{D_1}_{\text{real}} = 100$ and $a^{D_1}_{\text{real}} = 95$. The stochastic costs are therefore $C^s(S_k, \Theta^x_k, \omega) = \max(a^D_{\text{real}} - 95, 0) - \max(a^D - 95, 0) + \max(a^D_{\text{real}} - 100, 0) - \max(a^D - 100, 0) = 5 + 0 = 5$.

4 Solution Approach

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 multi-dimensional vectors, grows combinatorially 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. We are further challenged in that decisions must be made in real time, limiting the amount of time available for computation.

To address these challenges, we introduce a heuristic assignment policy for the RMDP that allows us to address the uncertainty while maintaining computational tractability in the face of large state and decision spaces. The parameterized policy has three main features:

Assignment Decisions To account for highly limited computation times, we implement a heuristic assignment policy based on the deterministic costs C^d.

Random Ready Times To account for the stochastic costs C^s, we implement a cost function approximation (CFA). In evaluating a decision, we incorporate a parameterized buffer that is added to the planned arrival times at customers. This buffer penalizes arrivals planned close to the deadlines and therefore guides the heuristic policy to favor more reliable assignment decisions.

Stochastic Orders To account for the uncertainty in orders, we allow the postponement of "not yet important" assignments. This postponement allows for the accumulation and potential bundling of requests and thus greater flexibility in later assignment decisions.

In the following, we first present the assignment heuristic. Then, we describe how we integrate the buffers and how we implement the postponement strategy. Finally, we present the algorithmic procedure summarizing the three features of our method. Due to the preparatory formulation of the problem as route-based MDP, the features can be described in an efficient manner.

4.1 Deterministic Costs: Assignment Heuristic

For the problem under consideration, the routing is determined by the drivers. Thus, each assignment results in a specific unalterable routing. Decisions are made only about the assignment of unassigned orders to drivers. 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 a set of o unassigned orders and h vehicles, the number of potential assignments is h^o without considering postponements. For four orders and 15 vehicles, this leads to $15^4 = 50,625$ potential assignment decisions. Keeping in mind that decisions are made in real-time within a few seconds, a consideration of every potential assignment solution is not feasible. Thus, we reduce the number of assignment solutions under consideration as follows.

First, we enumerate all potential ordered sequences of unassigned orders. Given o unassigned orders, this leads to a set of $o!$ sequences. In the example with four unassigned customers, this results in 24 sequences. 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 with the minimum 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. If 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 only consider $24 \times 4 \times 15 = 1,440$ 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 $C^d(S_k, \Theta_k, \Theta_k^x) = \Delta(S_k, \Theta_k^x) - \Delta(S_k, \Theta_k)$.

4.2 Stochastic Costs: Cost Function Approximation using Time Buffer

Decision making based only on the deterministic costs C^d ignores the impact of the random ready times. As shown in the motivational example in Section 3.2, even though the deterministic costs might be zero for all decisions, different decisions lead to different stochastic costs C^s. These stochastic costs result from ready times being later than planned. To account for these potential late ready times, we introduce a CFA. Powell (2017) 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, we base our CFA on a temporal buffer b.

This buffer is used as a penalty cost term to avoid "risky" arrivals close to the 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 $\Delta^b(S,\Theta)$ for state S and routing Θ is then defined as:

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

Deliveries with a planned arrival time ($a^D + b$) will contribute to the objective only if $a^D + b > t^D + \bar{t}$. A buffer greater than 0 will thus make it more likely a delivery will to contribute, albeit artificially, to the objective function since $a^D + b$ will be more likely to be greater than $t^D + \bar{t}$. A buffer of 0 leads to the original delay function $\Delta(S,\Theta)$, which depends on only deterministic costs. Further, a buffer of $\bar{t}$ leads to a planned delay of

$$\Delta^{\bar{t}}(S,\Theta) = \sum_{\theta\in\Theta}\sum_{D\in\theta} \max\{0, \underbrace{(a^D + \bar{t}) - (t^D + \bar{t})}_{=a^D - t^D}\}. \tag{5}$$

Delay as computed in Equation 5 results in the sending the closest vehicle to an order. We will use this buffer in one of our benchmark policies.

Revisiting the example in Section 3.2, a buffer of $b = 10$ minutes results in an planned delay of $\Delta^b(S,\Theta^x) = 5$ minutes and an assignment of Customer 2 to Vehicle 2 whereas the customer was assigned to Vehicle 1 originally. With the buffers, Customer 2 is assigned to Vehicle 2 for buffers $b > 5$. With the transition described in the example, this assignment also minimizes the stochastic costs.

Choosing the right buffer is challenging for several reasons. The buffer can be viewed as a trade off between efficient delivery and effective utilization of the available time before the deadline. A large buffer favors lower delivery times and reduces bundling opportunities. A smaller buffer may allow more bundling but may lead to delays. Further, a larger buffer may be necessary in case the variance in the ready times is high whereas a lower buffer is needed for instances with low variance.

To tune the buffers, we use sample average approximation (SAA) on the restricted policy class. See Fu (2015) for an overview of SAA. Let π^b be the described policy (including the assignment policy described above and the postponement strategy introduced subsequently) with buffer b. We define policies $b = 0, 2, \dots, \bar{t}$ minutes for each instance setting. For each instance setting, we run $2,000$ training runs $\Omega^1, \dots, \Omega^{2,000}$ and select the buffer minimizing the average delay. Mathematically, we have:

$$\pi^{\text{best}} = \underset{\{\pi^b: b=0,2,\ldots,\bar{t}\}}{\arg\min} \frac{1}{2,000} \left[\sum_{i=1}^{2,000} C^d \left(S_k^{\Omega^i}, \Theta_k, \Theta_k^{X^\pi(S_k^{\Omega^i})} \right) + C^s \left(S_k^{\Omega^i}, \Theta_k^{X^\pi(S_k^{\Omega^i})} \right) | S_0 \right],$$

where $S_k^{\omega^i}$ is the k^{th} state of training run ω^i.

4.3 Stochastic Orders: Postponement Strategy

Postponing assignment decisions may allow for more effective assignment. Hence, we incorporate postponements into our solution approach. 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 route. This means that we postpone only orders 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 according restaurant of the order. All other orders can be postponed without any immediate consequences in the route and may be later integrated in the route again. We allow orders to be postponed at most $t^p_{\max}$ minutes.

The number of postponements impacts the computational effort for the aforementioned assignment procedure. Computation time increases as the number of postponed orders increases. To allow real-time decision making, 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 are postponed.

4.4 Algorithmic Procedure

In this section, we summarize the algorithmic procedure for executing our solution approach. The procedure is represented by Algorithm 1. 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 of time t, the current routing Θ, and the unassigned orders $\mathcal{D}^o$. Other input parameters are the buffer b, maximal number of postponements $p_{\max}$, and maximal time of postponement $t^p_{\max}$. Output are the updated routing 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 $\mathcal{D}^x$. In a nutshell, the algorithm iterates through all ordered sets $\widehat{\mathcal{D}}$ of $\mathcal{D}^o$ and selects the one resulting in minimal delay. The algorithm considers the time buffers in the calculation of the delay. In case of several candidates with similar delay, the one with the largest average slack per customer is selected. For each routing candidate, the algorithm collect orders eligible for postponement. Eventually, it removes these orders from the routing and returns both the routing and the postponed orders.

The algorithm draws on the following functions:

FindVehicle$(\widehat{\Theta}, D, b)$ This function returns the vehicle of routing $\widehat{\Theta}$ 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$(\widehat{\Theta}, D, V)$ Given a routing, an order, and a vehicle, the function returns an updated routing with order D integrated in the route of vehicle V based on the driver's routing procedure.

Postponement$(\widehat{\mathcal{P}}, \widehat{\Theta}, D, p_{\mathbf{max}}, t^p_{\mathbf{max}})$ Based on the set of already postponed customers $\widehat{\mathcal{P}}$, the routing, and the two postponement parameters $p_{\max}$ and $t^p_{\max}$, this function returns a Boolean variable indicating whether this customer can be postponed.

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

4.5 Benchmarks

The proposed policy draws on both, postponements and a cost function approximation based on time buffers. We denote this policy *pbest*. We compare the policy to a series of benchmarks that are variations of *pbest*. In total, we consider four benchmark policies:

No buffers, no delay To analyze the impact of the combination of the delay and buffers, we test one policy with no buffers and no postponement of orders. We will refer to this policy in our experiments as $n0$.

Buffer, no postponement To isolate the impact of the buffers, 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 *nbest*. This policy is implemented by setting $p_{\max} = 0$. We note that we choose the best before for *nbest* in a manner analogous to the procedure we use for *pbest*. Thus, the time buffers 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 buffers. We refer to this policy as $p0$.

Base Policy: Current Practice To evaluate our policies, we also consider a base policy with a buffer of 40 and call the policy $n40$. With a buffer of 40, the policy seeks to send the vehicle that can serve the order fastest based on current information. In effect, this policy seeks to minimize, albeit myopically, the travel time associated with serving orders. This policy

Algorithm 1: Algorithm for the RMDP

```
Input : State $S$, Time $t$, Routing $\Theta$, Unassigned Orders $\mathcal{D}^o$, Buffer $b$, Maximal Number of
        Postponements $p_{\max}$, Maximal Time of Postponement $t^p_{\max}$
Output : Routing $\Theta^x$, Postponed Orders $\mathcal{P}^x$
1  // Initialization
2  $\Theta^x \leftarrow \Theta$// Best Routing
3  $\mathcal{P}^x \leftarrow \emptyset$// Best Postponements
4  delay $\leftarrow$ bigM// Delay
5  slack $\leftarrow 0$// Slack
6  // Assignment Procedure
7  for all $\widehat{\mathcal{D}}$ ordered set of $\mathcal{D}^o$ // All potential Sequences
8  do
9      $\widehat{\Theta} \leftarrow \Theta$// Candidate Routing
10     $\widehat{\mathcal{P}} \leftarrow \emptyset$// Set of Postponements
11     for all $D \in \widehat{\mathcal{D}}$ // All Orders in Sequence
12     do
13         $V \leftarrow \textit{FindVehicle}(\widehat{\Theta}, D, b)$// Find best Vehicle
14         $\widehat{\Theta} \leftarrow \textit{AssignOrder}(\widehat{\Theta}, D, V)$// Assign Order
15         if ($\textit{Postponement}(\widehat{\mathcal{P}}, \widehat{\Theta}, D, p_{max}, t^p_{max}) == \textit{true}$) // If Postponement
               Possible
16         then
17             $\widehat{\mathcal{P}} \leftarrow \widehat{\mathcal{P}} \cup \{D\}$// Postpone Order
18     end
19     if $(\Delta(S, \widehat{\Theta}) <$ delay$) \vee ((\Delta(S, \widehat{\Theta}) ==$ delay$) \wedge (\textit{Slack}(S, \widehat{\Theta}) <$ slack$))$ // If
           Best Routing
20     then
21         $\Theta^x \leftarrow \widehat{\Theta}$// Update Routing
22         $\mathcal{P}^x \leftarrow \widehat{\mathcal{P}}$// Update Postponements
23         delay $\leftarrow \Delta(S, \widehat{\Theta})$// Update Delay
24         slack $\leftarrow \textit{Slack}(S, \widehat{\Theta})$// Update Slack
25     end
26 end
27 $\Theta^x \leftarrow \textit{Remove}(\Theta^x, \mathcal{P}^x)$// Removed Postponed Orders and Restaurants
28 return Routing $\Theta^x$, Postponed Orders $\mathcal{P}^x$
```

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 Experimental Design and Implementation

In this section, we present the experimental design and analysis. We first define the instance settings based on a restaurant meal delivery operation currently existing in Iowa City. We then describe the details of our implementations of the proposed policy and benchmark policies. We next define the measures to analyze the impact of the policies on the different stakeholders. Finally, we describe the results of an extensive computational analysis.

5.1 Instance Settings

Our experiments represent the delivery of restaurant meals in Iowa City, a city whose metropolitan statistical area has an estimated population of 168,828 people in the 2016 (U.S. Census Bureau, 2017). Iowa City is a 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. Customer orders follow a Poisson process over time leading to uniformly distributed requests during the order capture phase. Conversations with local providers suggest that this assumption matches the request pattern of Iowa City. The deadline per order is $\bar{t} = 40$ minutes.

We consider 110 restaurants, based on the summer 2016 list of restaurants available from a meal delivery company operating in Iowa City. The locations of the restaurants are shown in Figure 2. Most of the restaurants are either in downtown Iowa City or in suburb of Coralville. We observe several restaurant clusters that offer potential for bundling orders, but also single restaurants distributed within the area. The time to prepare meals at each restaurant is assumed Gamma distributed with a mean of 10 minutes. We choose this 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.

We consider a fleet of 15 vehicles originating in different parts of the service area. The order capture phase is $T = [0, 420]$ minutes. This usually leads to working hours of less than 8 hours

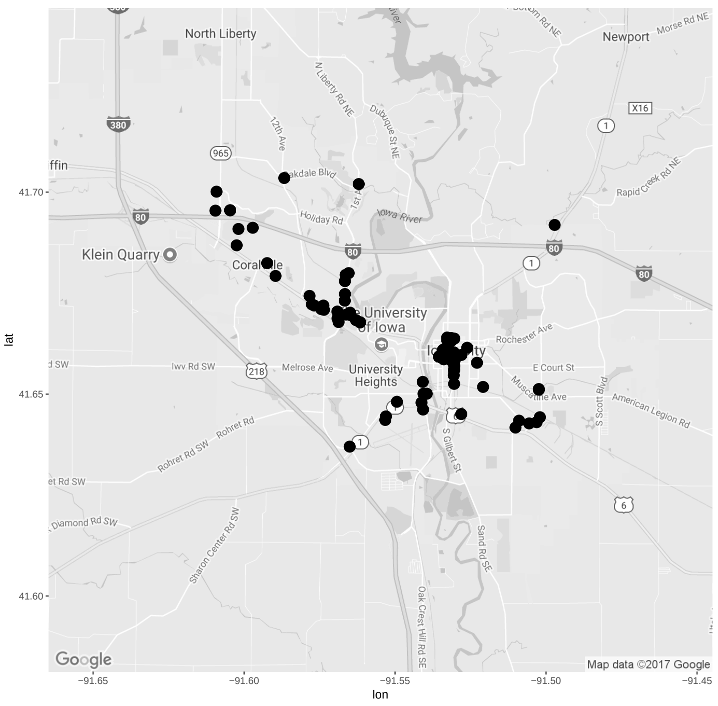


Figure 2: Restaurant Locations

for the drivers of these vehicles. The Euclidean distance between restaurants and customers are computed. To simulate a street network, we multiply these distances by a factor of 1.4 (Boscoe et al., 2012). The travel speed over these distances is 40 kilometers per hour. The service time at a customer and at a restaurant, once the food is ready, is 2 minutes.

To analyze the impact of random orders, 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 180, 240, and 300 expected total orders. Second, to account for stochastic meal preparation times, we vary the coefficient of variation (CoV) between $0.0, 0.1, \dots, 0.6$. 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. For each instance setting, we do 2000 simulation runs with different customer orders and sequences. These 2000 simulation runs are repeated five times, once for each policy under consideration ($n0$, $nbest$, $n40$, $p0$, and $pbest$). This creates a total of 210,000 runs with about 70 million decision points. 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 128GB RAM. 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.

5.2 Implementation

In this section, we describe the implementation details of the proposed policy and benchmark policies. For both $pbest$ and $nbest$, we tune the buffers with 2000 tuning runs as described in Section 4.2. We set $p_{\max}$ and $t^p_{\max}$ based on preliminary tests. For all instances, we set the maximum time of postponement to $t^p_{\max} = 30$ minutes. We note that the policies are nearly invariant for $t^p_{\max} > 15$. For instances with 180 and 240 expected orders, we set the number of postponements to $p_{\max} = 3$. For instances with 300 expected orders, we set $p_{\max} = 2$ due to computational limitations.

Implementing all of our policies also requires a method to route and reposition drivers. In practice, drivers are independent contractors. Thus, their routing and repositioning decisions cannot be controlled, but we can make certain assumptions. First, we assume that no drivers decline the orders they are assigned by the policies. Next, we assume that the drivers follow a mobile navigation device to determine the best paths between customers and/or restaurant. We approximate the travel time of these paths using the Euclidean distance and the factor of 1.4 as discussed above.

If the driver has at least one unserved order when he or she receives a new order, we assume that the driver determines the best sequence to serve the bundle using a generalization of the Cheapest Insertion-method (CI) by Rosenkrantz et al. (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. In that case, the procedure maintains this 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. The restaurant is called "empty" if no other driver is currently idling or on the way to that restaurant.

5.3 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. In our experiments, for each measure, we compute the average value from 2000 simulation runs for each instance setting and policy.

Customers: Delay

Customers want reliable and fast service, so we measure the average total delay as in our MDP objective. Since long delays create the customer dissatisfaction and can impact retention of these

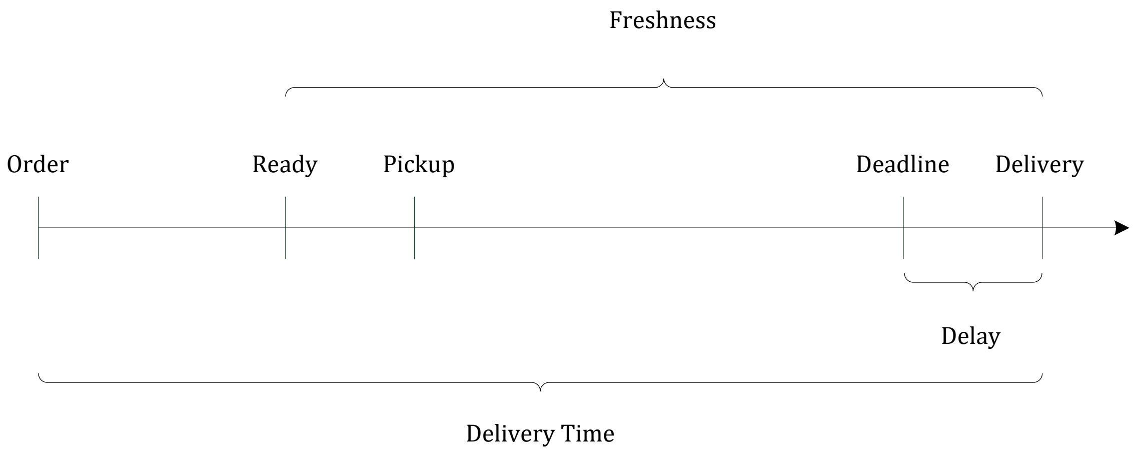


Figure 3: Measures

customers. 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 3.

Restaurants: Freshness

To satisfy their customers and grow their business, restaurants need their food to be fresh when it arrives at the customers. We define two measures: the average freshness of the food and the average maximum un-freshness. The average freshness is the average time passed between ready time of an order at a restaurant and delivery time at the customer. The average maximum un-freshness is computed based on the maximum time passed between ready time and delivery time in the course of a day/simulation run.

Drivers: Number of Orders

The drivers' objective is to make as many deliveries as possible to maximize his or her wages. Because the orders are not always divided evenly among the 15 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 spans 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.

6 Computational Results

This section presents the results of our computational analysis. We report the results focusing on customers, restaurants, and finally drivers, the three stakeholders whom delivery companies must satisfy to grow their businesses.

6.1 Customers

For the customers, we analyze the policies' impact on average total delay, maximum delay, and delivery times.

Average Total Delay

In this section, we discuss the computational results demonstrating the effectiveness of the proposed policy versus the benchmarks. All experiments in this section assume homogeneous distributions on the time to prepare food. In Appendix A, we present results for heterogeneous distributions. The results lead to the same conclusions as are presented here.

First, we compare the average reduction in total delay of the policies $pbest, nbest, p0$, and $n0$ as compared to the base policy $n40$. The reduction is defined by the difference in value of the total delay from $n40$ and value of the particular policy divided by the total delay from $n40$. We calculate the average reduction for the 2000 runs per instance setting and then calculate the average over these instance settings. The results are shown in Figure 4.

We observe that all policies, on average outperform the base policy significantly. Policy $pbest$ 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 to the other polices. The use of the parameterized buffer alone has a higher impact on the reduction than the postponement. As we will show next, the value of adding the parameterized buffer and/or postponement varies with respect to the instance settings' characteristics. It is also interesting to see the improvements gained in total delay with $n0$. This shows that even a policy with deterministic costs and no postponements can reduce delay as compared with our base policy.

The results for the different instance settings are shown in Figure 5. The three rows represent the expected total number of orders (180, 240, 300) and the columns represent the CoV in the meal

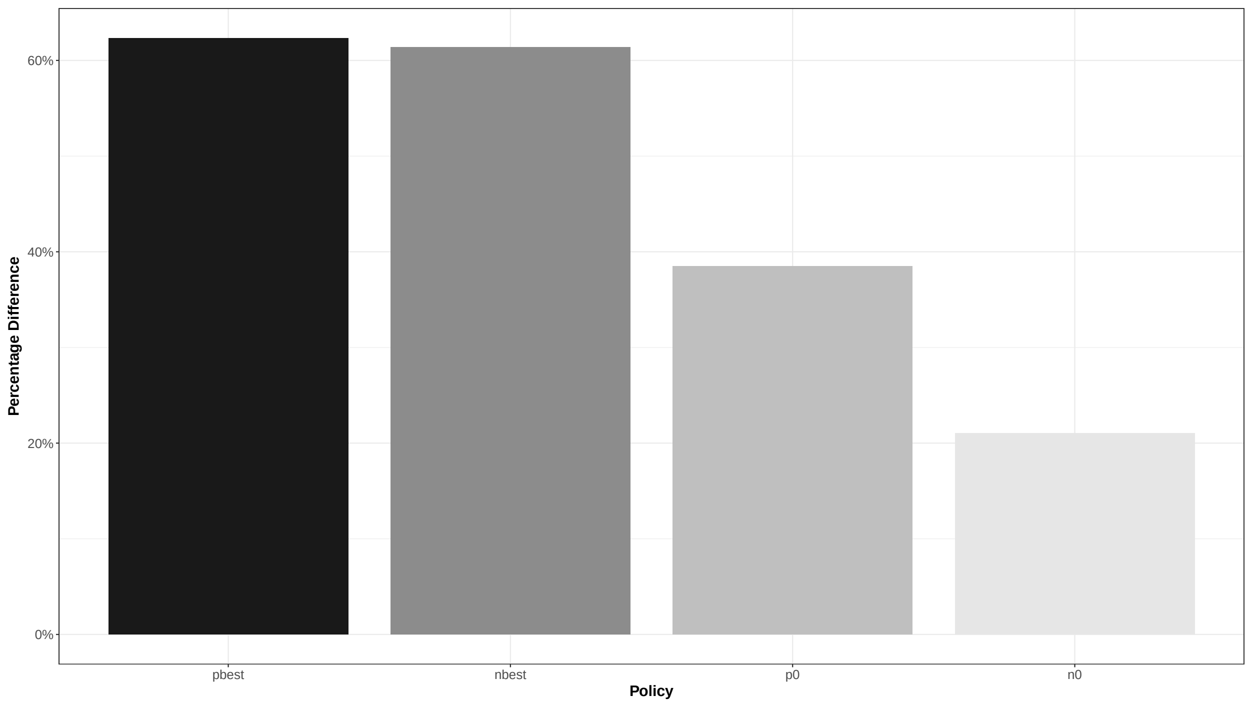


Figure 4: Overall Improvement in Total Delay as Compared to Policy $n40$

preparation times. The individual bars represent the percentage decrease in total delay for $n0$, $p0$, $nbest$ and $pbest$, as compared with the results found with $n40$. We observe that the importance of buffer and postponements vary based on structure of the instance setting. For example, we see *increases* in total delay for $n0$ and $p0$ with 180 expected orders and CoV ranging from 0.3 to 0.6. With 180 expected orders, positive improvements in total delay consistently occur with $nbest$ and $pbest$ but particularly with low CoV in meal preparation times. These results indicate that our ideas are not particularly helpful with a low number of orders and a high CoV, but can be helpful with lower CoV in meal preparation times. We can also see that our ideas become more impactful when the number of orders increases. For instances with 240 expected orders, the only increases in total delay come with $n0$ for CoV of 0.5 and 0.6 and $p0$ for a CoV of 0.6. All of the other instance settings with 240 expected orders show a decrease in total delay. Interestingly, for most CoV values, $nbest$ and $pbest$ continue to perform the best, but there is not a big difference in the savings in total delay from the two policies. This indicates that, with 240 expected orders, offering postponement plus buffer does not offer much difference than only offering a parameterized delay. For instances with 300 expected orders, the savings in total delay from $n0$, $p0$, $nbest$, and $pbest$ become much more similar. This result occurs because, with more expected customers, the drivers are all utilized almost all of the time. Correspondingly, we will later show that, with increasing number of orders, the best buffers get smaller. We further observe that the savings in total delay is still larger for low

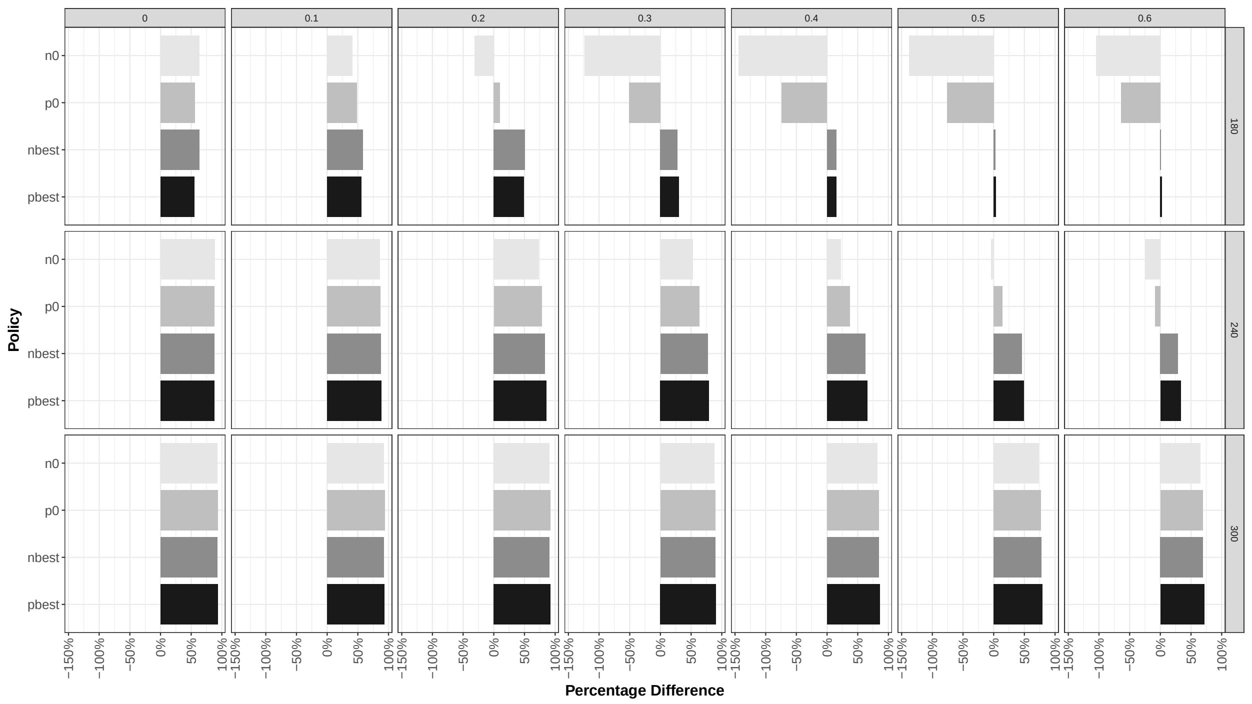
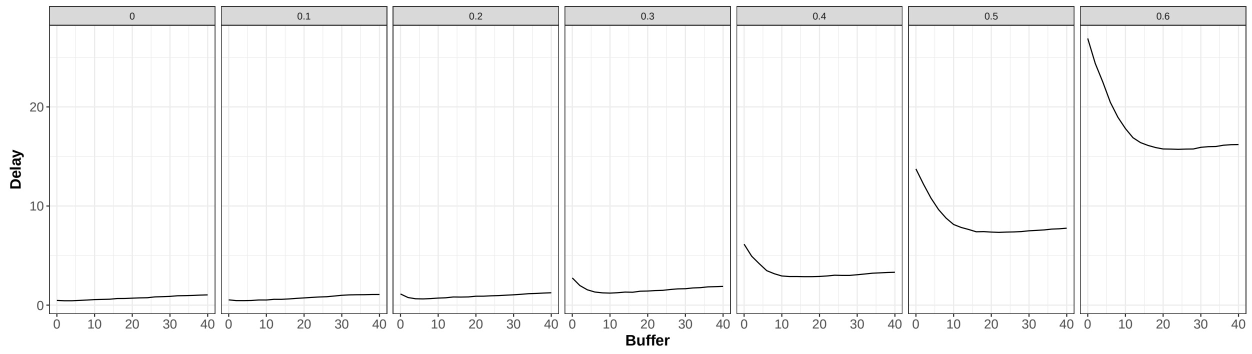


Figure 5: Percentage Difference in Total Delay Compared to Policy $n40$ by Expected Number of Customers and CoV

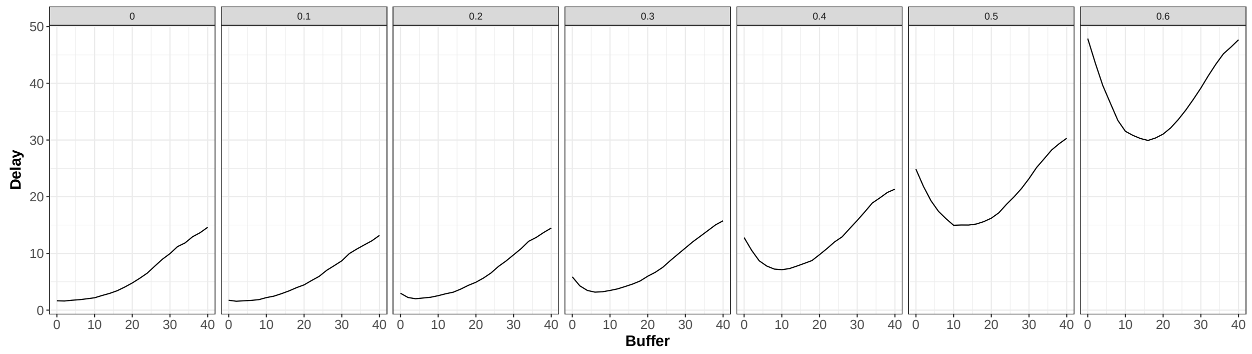
CoV than high CoV, but now all values are positive.

We now analyze how the buffers impact the total delay. Figure 6 shows the total delay for various buffers when using postponement. Figure 6a shows the development for 180 expected orders, Figure 6b for 240 expected orders, and Figure 6c for 300 expected orders. Overall, we observe that the total delay increases with both the number of orders and CoV. This is not surprising, because more orders lead to a larger workload and a higher CoV leads to more uncertainty and variance in the ready times. For 180 expected orders, Figure 6a shows large buffers achieve the smallest total delay for each CoV. Keeping in mind that a buffer of 40 translates to a policy minimizing total delivery time, these results indicate that, on days with few orders, sending the closest vehicle may be a suitable policy. This is especially the case when the CoV is high. For 240 expected total orders, Figure 6b shows that the best solution quality is achieved with moderate buffers. As we show later in this section, there is an increase in the number of bundling operations performed with a smaller time buffer. For 240 expected orders, we experience a tradeoff between the flexibility of bundling and delays. Finally, for 300 expected orders, Figure 6c demonstrates that the best buffers are always rather small. For these instance settings, the deadlines need to be exploited as much as possible to bundle and to maintain flexibility for future orders.

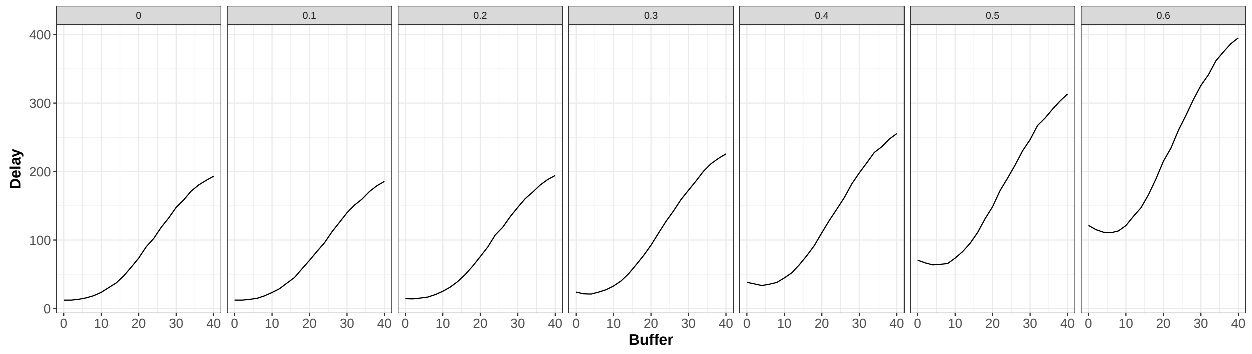
Another of the reasons for the performance of the various policies is the degree to which they



(a) 180 Expected Orders



(b) 240 Expected Orders



(c) 300 Expected Orders

Figure 6: Average Total Delay for Policies per Instance Setting

allow for bundling operations. Figure 7 presents the average number of bundling operations over all instance for each of the five policies $pbest$, $nbest$, $p0$, $n0$, and $n40$. We observe that the number of bundling operations for the base policy $n40$ is significantly smaller than for the other policies. The policy $n40$ seeks to minimize delivery time and thus always uses the fastest vehicle available. In assigning the fastest vehicle, there is less opportunity for the accumulation of customers that can be bundled. With that in mind, it is not surprising that policies incorporating postponement lead to more bundling than when a policy does not postpone. Postponements enable future bundling

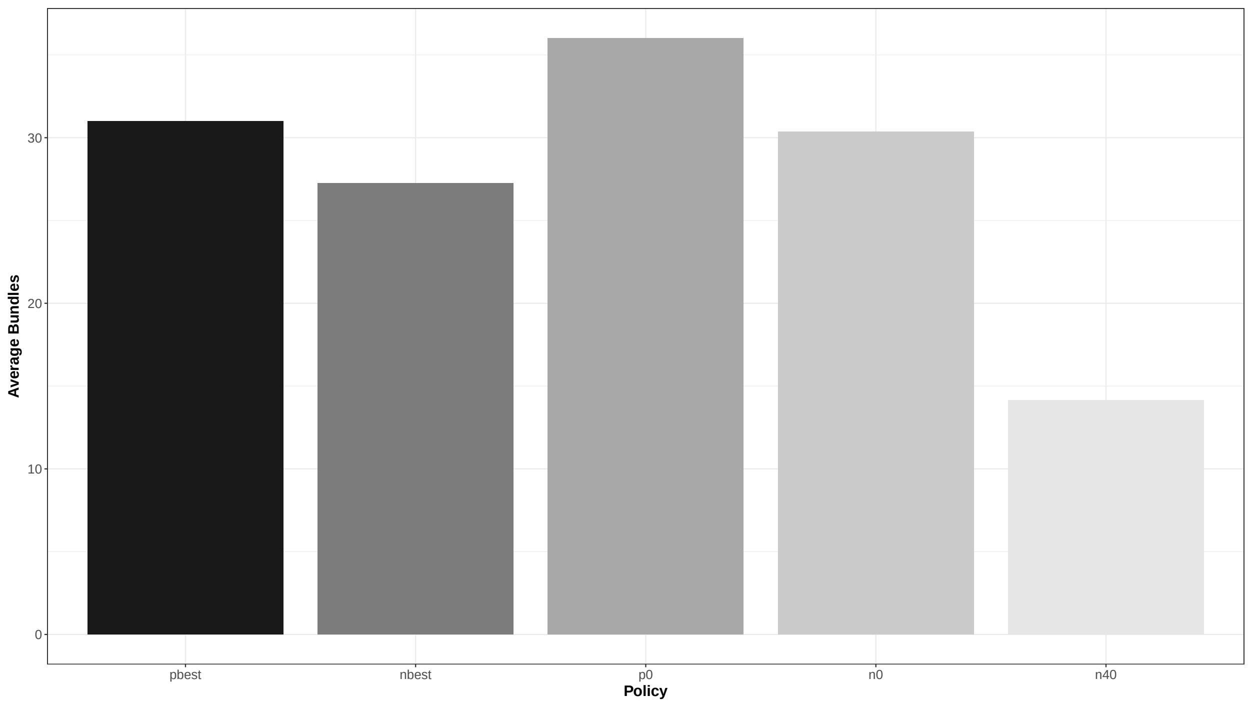


Figure 7: Average Number of Bundling Operations

opportunities. Further, there is a dependency between buffers and bundling operations. We see that $pbest$ and $nbest$ average fewer bundling operations than $p0$ and $n0$, respectively. Figure 8 demonstrates the effect in detail on the instance setting of 240 expected orders and a CoV of 0.2. In Figure 8, the x-axis shows the buffer. The y-axis shows the number of bundling operations. Buffers of 0 achieve the most bundling operations. As the buffer value increases, policy more closely mirrors the minimization of delivery time, and the number of bundles decreases.

We note that the previous discussion assumes bundling to be allowed. Appendix B presents results of policies in which we do not allow any bundling. The results show that bundling improves solution quality particularly when the CoV is lower.

Average Maximum Delay

In addition to total delay, we consider the impact of our policies on the maximum delay. This analysis is important as minimizing total delay may place the delay burden onto a small number of customers. Figure 9 shows the improvement of the proposed policy and benchmarks with the respect to the base policy $n40$ averaged across all instance settings. These improvements are shown in Figure 9. The axes are analogous to those in Figure 4 as are the results. Policies $pbest$ and $nbest$ improve the maximum delay as well as the average delay, and again $p0$ and $n0$ create improvements but not as much as with parameterized buffers. These results indicate that the reduction in the

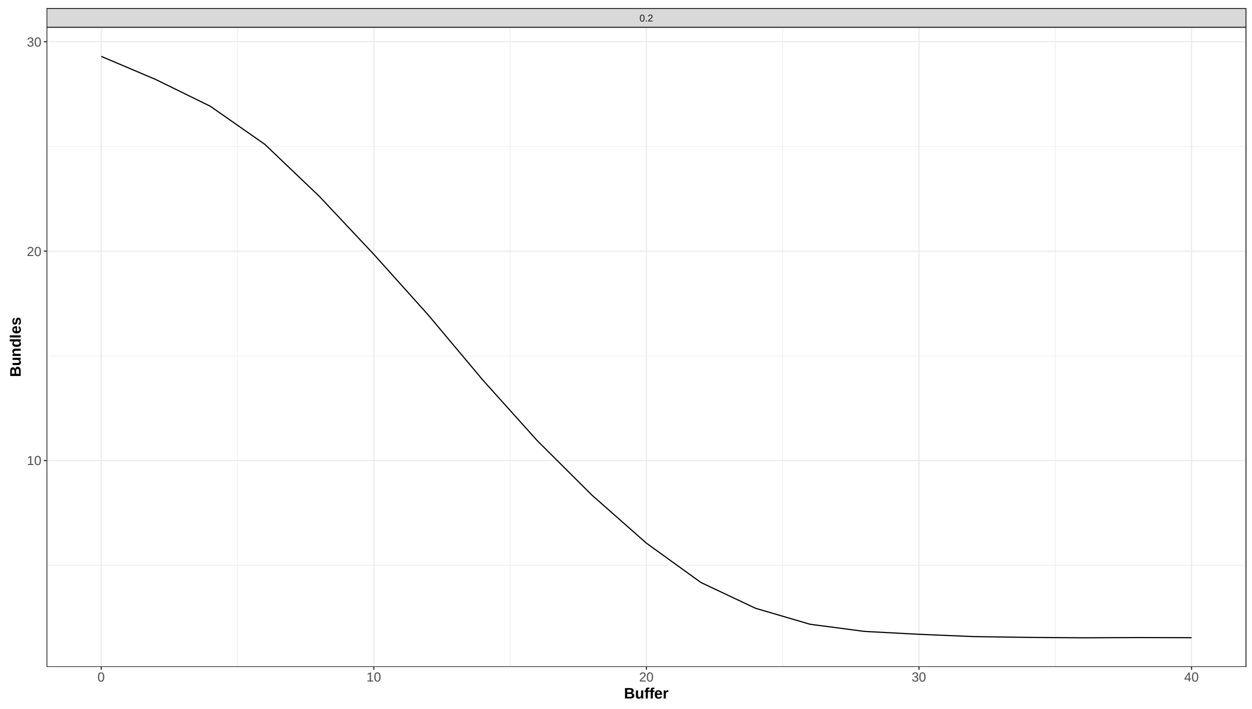


Figure 8: Average Number of Bundling Operations by Time Buffer Size for Instance with 240 Customers and CoV of 0.2

average total delay for these policies is not obtained by good service to a majority in exchange for poor service for a minority of customers. Overall, our policy $pbest$ improves customer experience on average and also in the worst case significantly.

Delivery Times

We complete our analysis related to customer service by analyzing how minimizing delay differs from a policy that seeks to minimize the overall delivery time, where delivery time is defined as the time between when a customer places an order and its delivery. To do so, we compare the proposed policy $pbest$ to the base policy $n40$. For each policy, we record the number of orders that have delivery times between 0-5 minutes, 5-10 minutes, . . ., to 75-80 minutes for each simulation run and average them for each instance setting. We then compute the difference in these values (average from $pbest$ - average from $n40$) between the two policies, Thus, in case of a positive value, $pbest$ conducts more services with these delivery times than $n40$. Figure 10 presents the results by expected number of orders and CoV. In Figure 10, the x-axis indicates the delivery time range, 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 $pbest$ serves fewer customers fast (e.g. under twenty minutes), but also fewer customers late (more than 40 minutes). Policy $pbest$ concentrates

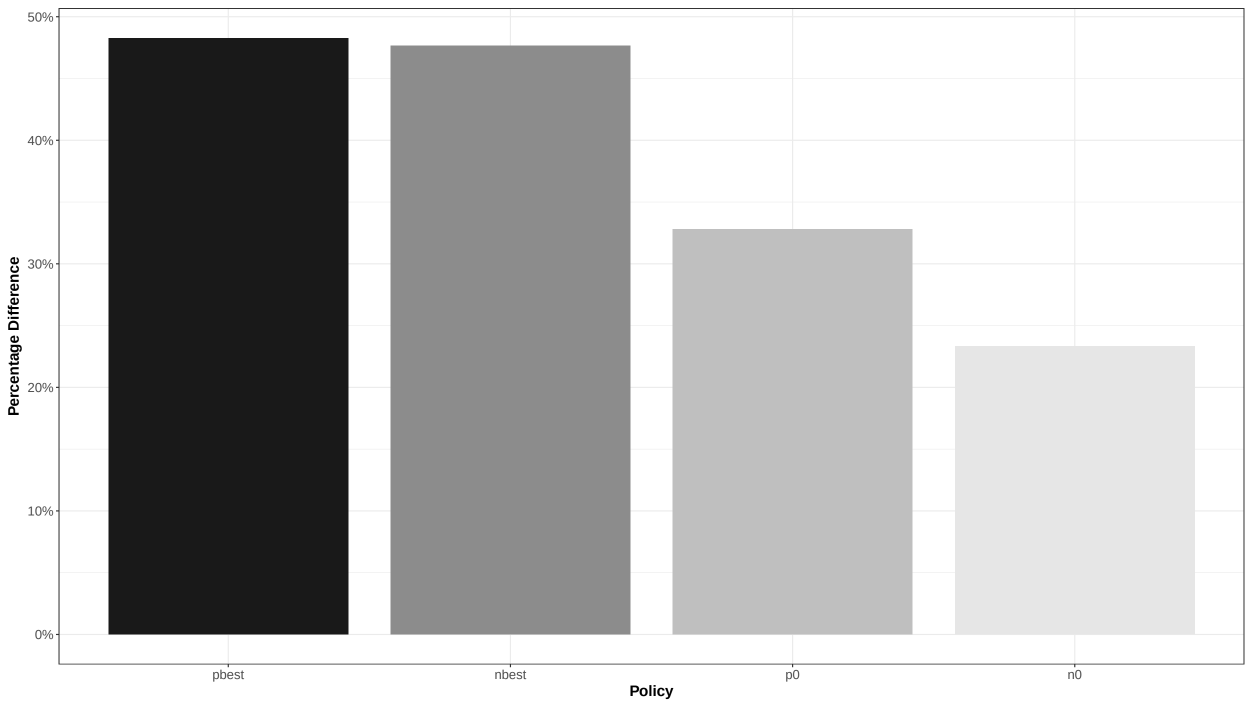


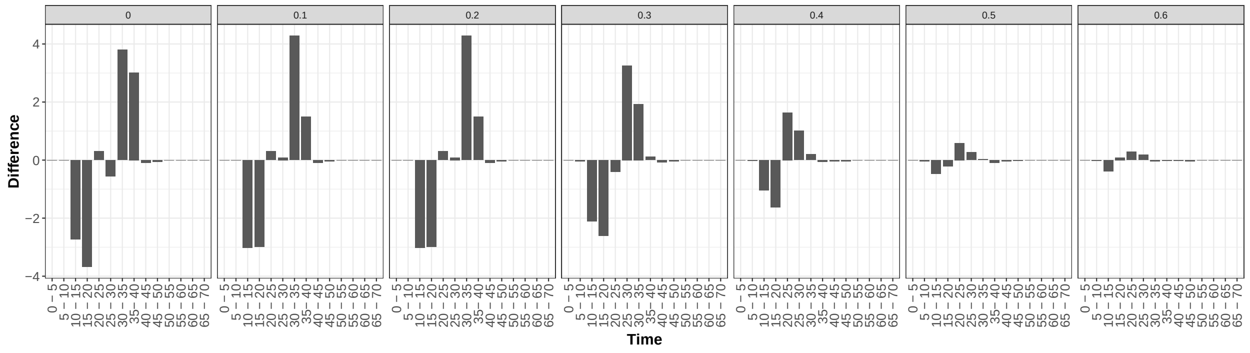
Figure 9: Overall Reduction in Average Maximum Delay Compared to Policy $n40$

deliveries with delivery times between 30-40 minutes with 180 expected total orders, 25-40 minutes with 240 expected total orders, and 15-35 minutes with 300 expected total orders. As can be seen in Figure 6, the concentration of deliveries in these time intervals is reflective the best time buffer in each respective case. Thus, the effect of the buffer is to reduce the number of fast deliveries while also reducing the number of late deliveries. This is a trade off that customers are likely to be happy with.

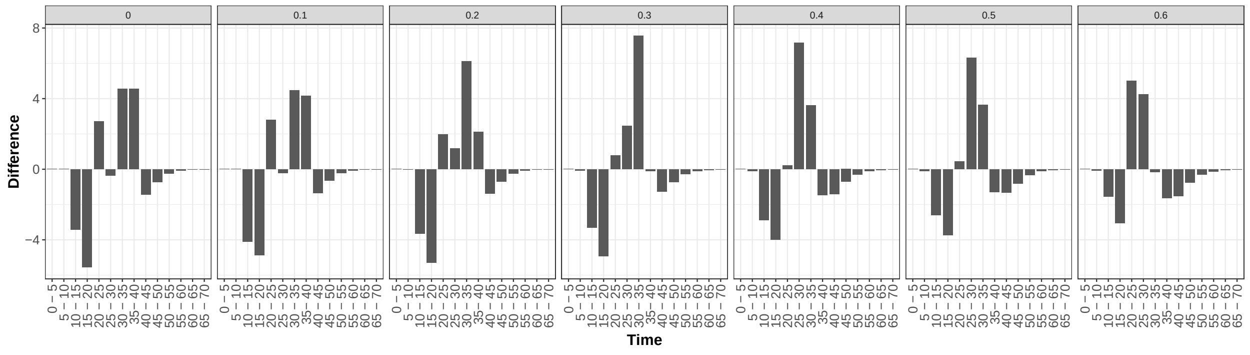
Interestingly, these results are consistent across the range of CoV values. The most noticeable effect of increasing CoV is that the differences diminish. This effect is most notable in the 180 expected orders case. In that case, the two policies are converging as is shown by the increasing buffers in Figure 6a. The convergence effect is minimal in the case of 300 expected orders. This result too is reflected in the buffers seen in Figure 6c, where the buffers are much smaller by CoV relative to the case of 180 expected orders. This difference reflects the uncertainty associated with the timing of the orders themselves when the number of orders increases.

6.2 Restaurants

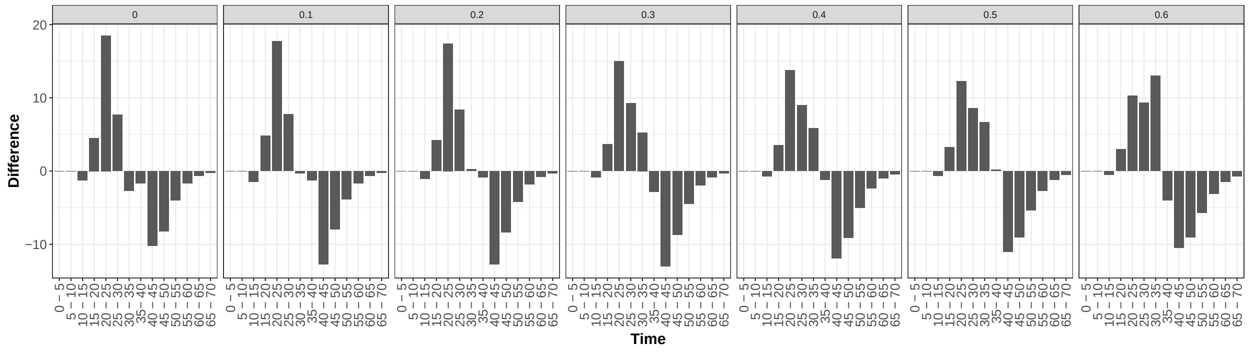
We now analyze how our proposed policy and the benchmark policies impact the restaurants. Restaurants seek good reviews and returning customers. In addition to the timely arrival of the food



(a) 180 Expected Orders



(b) 240 Expected Orders



(c) 300 Expected Orders

Figure 10: Difference in Orders per Delivery Time Range for $pbest$ and $n40$

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), while 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 both the average freshness of the food and the maximum un-freshness experienced by customers.

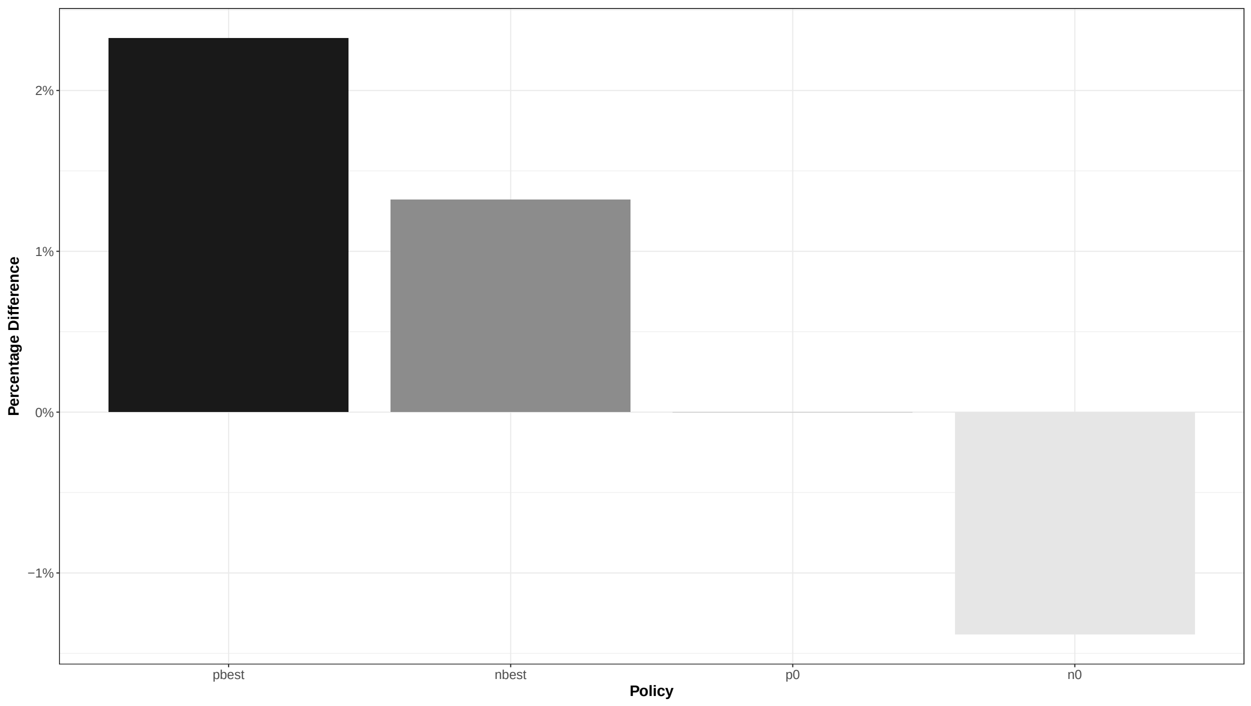


Figure 11: Improvement in Freshness as Compared to Policy $n40$

Total Freshness

Figure 11 shows the improvement of the policies in average freshness relative to the base policy $n40$. Both $pbest$ and $nbest$ improve upon the base policy. Yet, relative to the delay measure, the improvements of the policies relative to $n40$ is much smaller in this case. The improvement of policy $pbest$ is only 2.3%. The average freshness decreases with policy $n0$. The reason for the reduced performance gains and even loss is that the freshness is not surprisingly strongly connected to the delivery time, and $n40$ myopically minimizes delivery time. Thus, the base policy $n40$ does relatively well itself in terms of delivering fresh food on average.

However, the aggregate results do not tell the whole story. Figure 12 shows the results for the percentage difference of the proposed and benchmark policies compared to the base policy for each of the expected number of customers. The figure demonstrates that there is a strong correlation between the performance of the policies and expected number of customers. Because $n40$ sends the fastest vehicle, the policy performs well for smaller numbers of orders. The performance of policy $n40$ decreases as the number of expected orders increases. This result reflects the fact that, while the fact that policy $n40$ minimizes delivery time and thus unfreshness, its myopic nature reduces quality as the number of customers increases and their random arrivals increase. Thus, with 300 expected orders, the improvement of $pbest$ over the base policy is over 10% in terms of freshness.

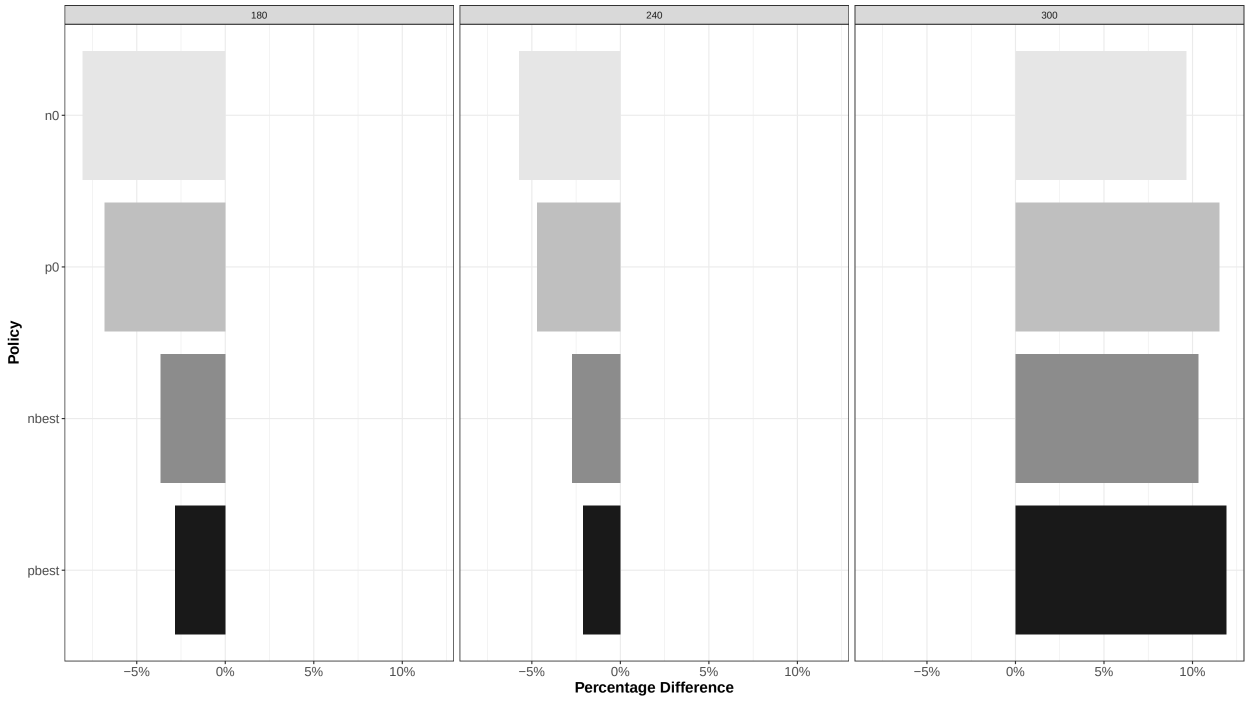


Figure 12: Improvement in Freshness Compared to Policy $n40$ by Expected Number of Customers

Average Maximum Unfreshness

Figure 13 shows the improvement of the policies in maximum un-freshness as compared with the policy $n40$. The results are strongly connected to the maximum delay. Here, the improvement of all policies is high with a value for $pbest$ of more than 15%. Further, Figure 14 shows that $pbest$ outperforms the base policy at all order levels. Thus, with $pbest$ particularly, the worst average customer experience is reduced and significantly so when the expected number of customers is 240 or more.

6.3 Drivers

In this section, we analyze how the policies impact the drivers. We first analyze the assignent of the orders to the drivers in the short and medium term. We then analyze how the policies provide either higher solution quality with fewer vehicles or to grow business by serving more orders with the same fleet size. Both cases benefit not only the company but also the drivers who serve more orders when they are fewer vehicles or more orders.

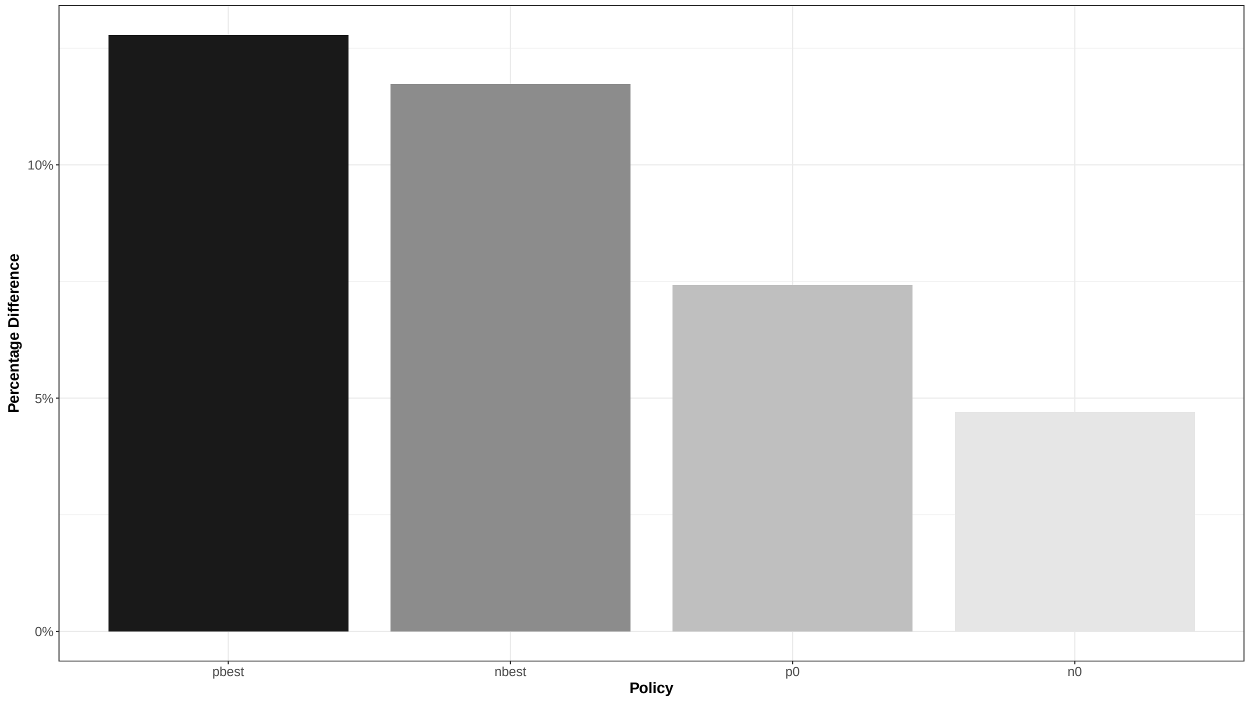


Figure 13: Improvement in Maximum Un-Freshness Compared to Policy $n40$

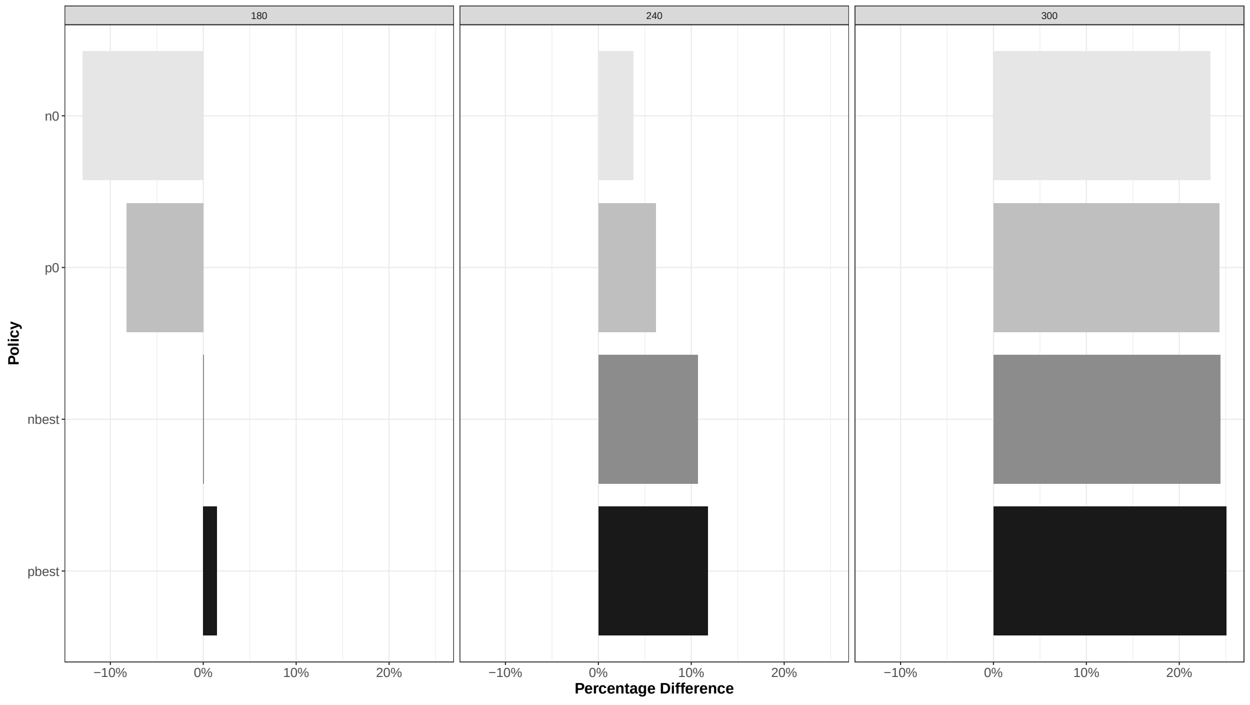


Figure 14: Maximum Un-Freshness Compared to Policy $n40$ by Expected Number of Customers

Average Minimum and Maximum Number of Orders Served Per Day and Over 20 Days

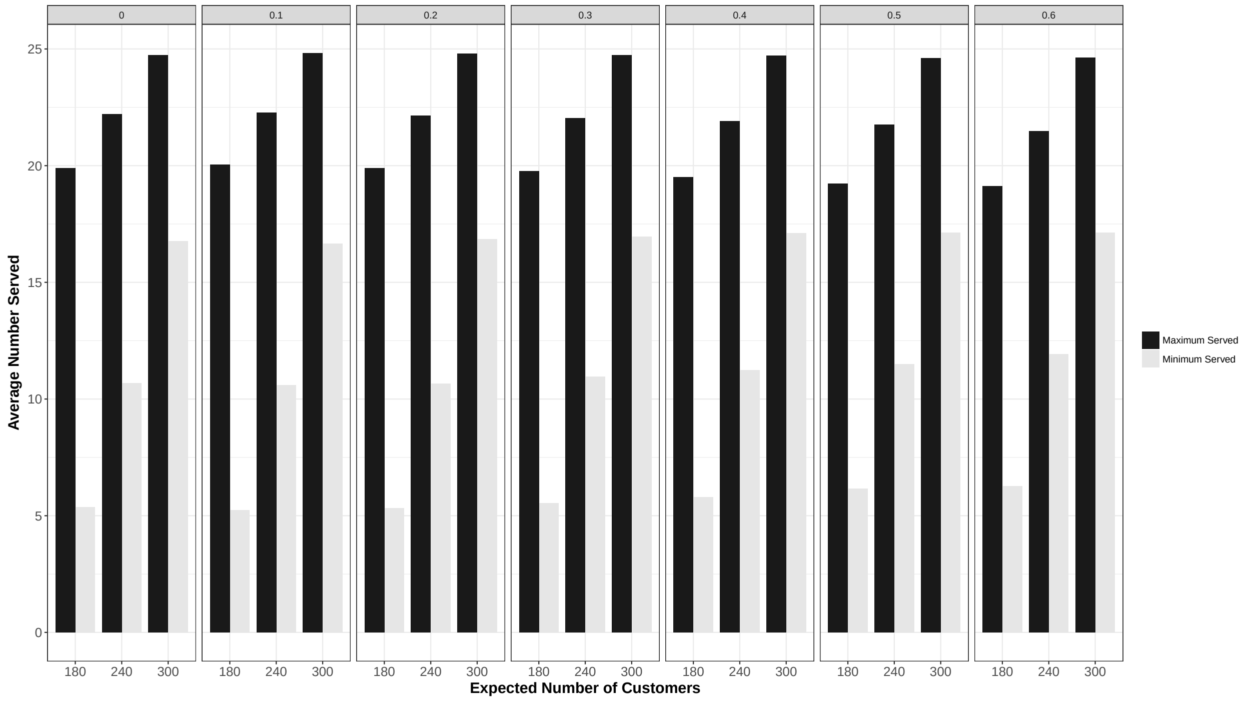
Bundling operations have an impact on the assignment decisions for the drivers. Since drivers are paid based on the number of orders they deliver, some policies may be preferred by drivers. In this section, we analyze how the policies change the number of assignments to drivers, as measured by the maximum number of customers served by any driver and the minimum number of customers served by any driver. If the disparity is sufficiently large, drivers will interpret the assignment policies as favoring some drivers over others, affecting driver retention.

Figure 15a shows the average minimum and maximum number of orders served per day by drivers when policy $pbest$ 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 with the lowest number of expected orders. The reason for these disparities is that when a driver gets near a cluster of busy restaurants, it becomes advantageous to bundle and to assign that driver to additional orders in that area.

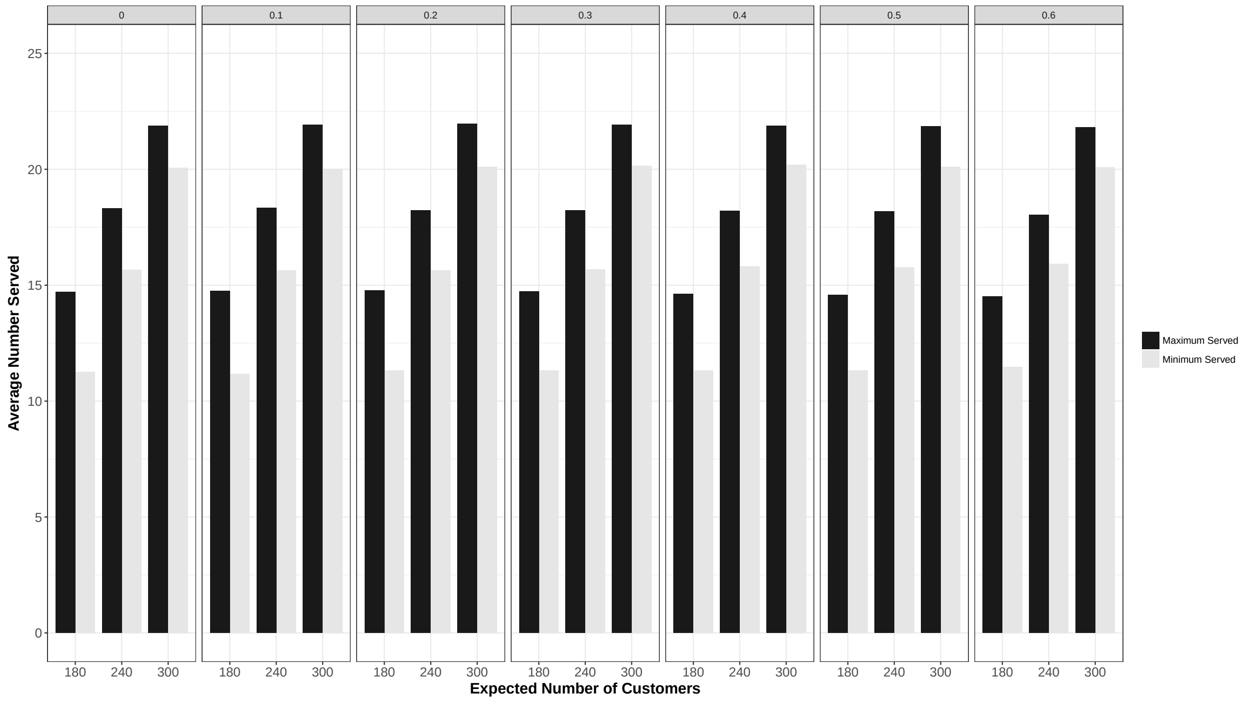
Figure 15b presents analogous results to those in Figure 15a, but first averages each driver's number of customers served over 20 days or approximately one month of work days. To compute the 20-day values, we use our existing runs and batch them into 20-day groups. For each set of 20 out of our 2,000 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. When compared to Figure 15a, Figure 15b show that over time the disparities between the orders served by the driver serving the maximum and the driver serving the minimum are greatly decreased. 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.

While the effect over 20 days may average out, there could still be policies that are preferable to $pbest$ in terms of fairness. To compare fairness, for each policy, we compute the difference between the maximum number served and the minimum number served. We then take subtract the two differences. A positive value indicates that the application of $pbest$ results in greater difference between the maximum and minimum number of orders served by drivers and more inequity than $n40$.

Figure 16 presents the results. The global x-axis shows the number of orders. The local x-axis depicts the CoV. The y-axis depicts the average daily and 20-day differences. On a daily basis, we observe differences of up to 1.5 customers per day. Because of the increase in bundling operations with policy $pbest$, a few drivers may serve more and a few drivers may serve less customers on a day than others. In general, we also observe a decrease in difference when the CoV increases because then the policies become more similar.



(a) Average Minimum and Maximum Customers Served by Driver for Policy $pbest$



(b) Average Minimum and Maximum Customers Served over 20 Days for Policy $pbest$

Growing the Business While Offering the Drivers More Opportunities

While fairness is important, drivers also want increased opportunities. Relatedly, the company wants to grow its business and do so while maintaining its customer service level. The proposed

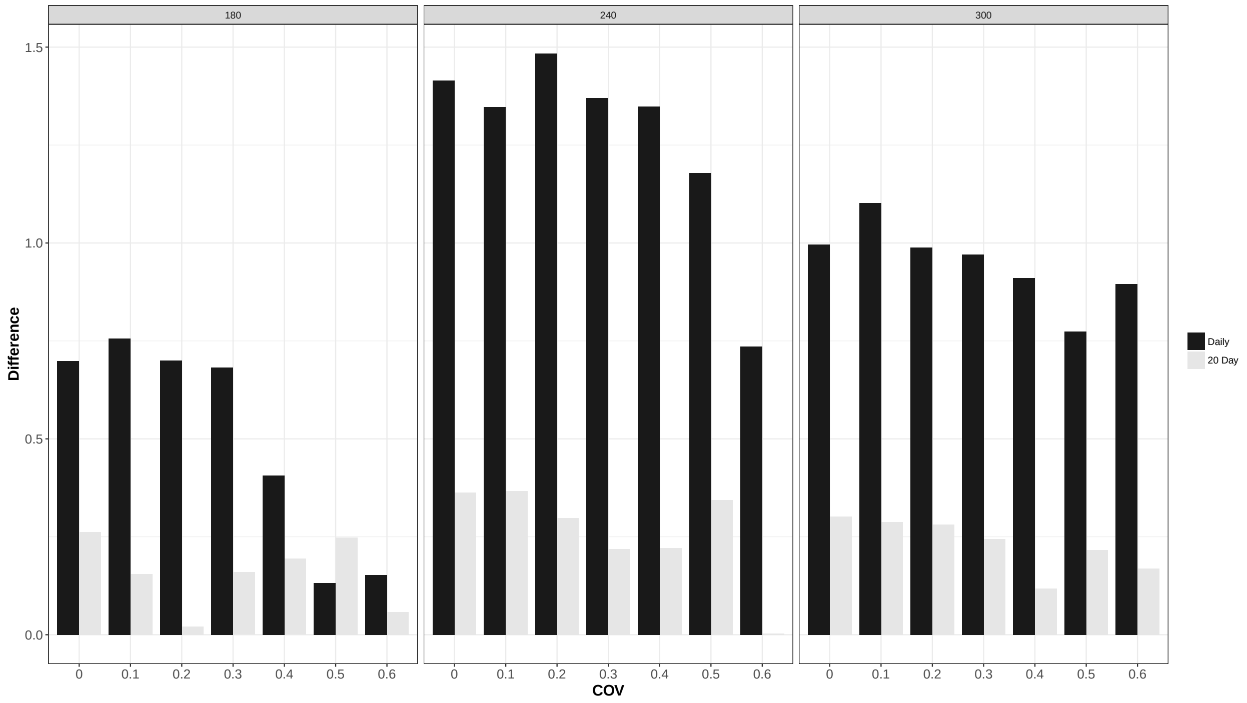


Figure 16: Difference in Driver Fairness between Policies $pbest$ and $n40$ per Day and over 20 Days

$pbest$ policy offers this opportunity. We demonstrate this in two ways. First, we show that $pbest$ can provide a higher level of customer service using fewer vehicles than the base policy $n40$. It is important to note that the policy $n40$ represents a policy of assigning the fastest driver to a request. Second, we show that the policy $pbest$ can serve more customers without sacrificing service quality than $n40$.

To demonstrate our first point, we focus on the instance setting with 240 expected customers and CoV of 0.2. We vary the number of drivers between 12 and 20 and analyze how the solution quality changes for policies $pbest$ and $n40$. We benchmark the solution quality with the setting of 15 vehicles, 240 expected customers, and CoV of 0.2 using policy $n40$. The results are shown in Figure 17. The x-axis shows the number of vehicles. The y-axis shows the improvement with respect to 15 vehicles, 240 expected customers, and CoV of 0.2 using policy $n40$. 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 $n40$ is not able to achieve the solution quality of policy $pbest$ and 15 vehicles. For policy $pbest$, even with only 14 vehicles, the solution quality still outperforms $n40$ with 15 vehicles. These observations are important because by providing the same solution quality with fewer drivers, $pbest$ allows a company to provide more opportunities for drivers without affecting customers.

While the meal delivery company needs to retain drivers to be successful, its ultimate goal is

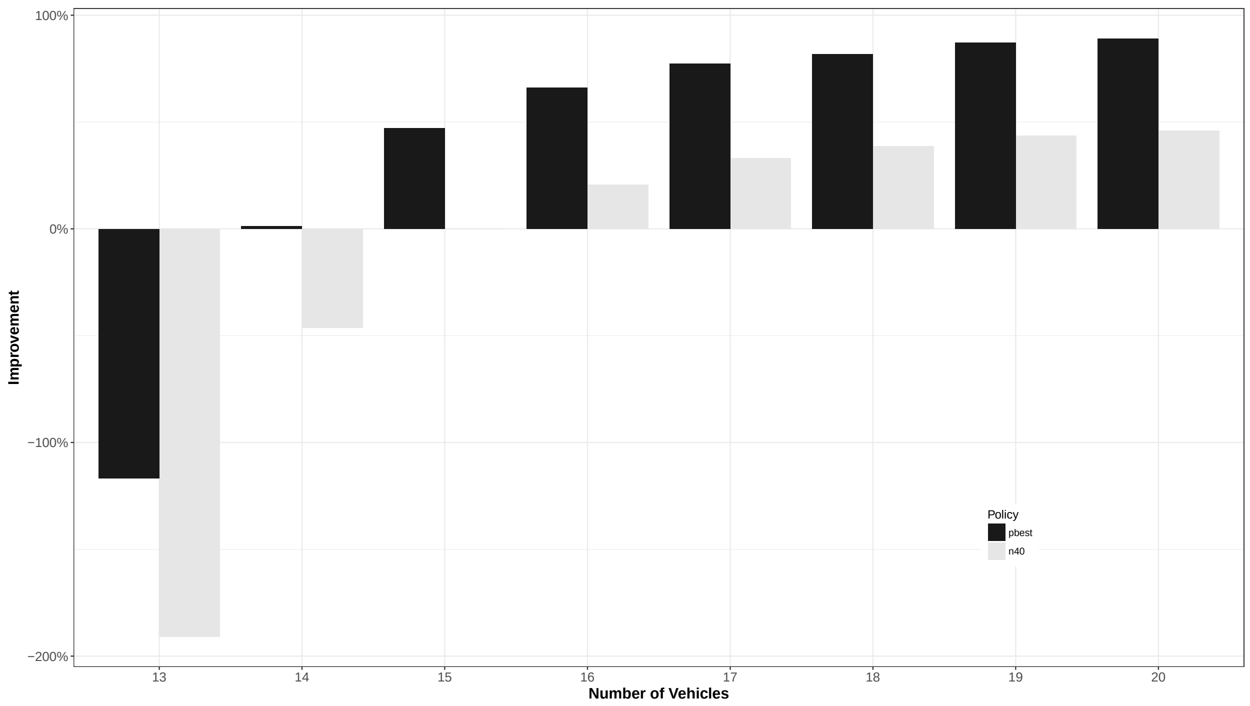


Figure 17: Vehicles Needed by Policies $pbest$ and $n40$ to Provide Comparable Solution Quality

to grow the business. We next show that the proposed policy $pbest$ can facilitate this growth by serving more customers than policy $n40$ while providing better customer service. To this end, we compare the instance settings with CoV of 0.2 with both 240 and 300 expected customers. For the instance setting with 240 expected customers, we apply policy $n40$. For the instance setting with 300 expected customers, we apply policy $pbest$. Comparing the total delay, the value for policy $pbest$ is 10.0% higher. Yet, $pbest$ is on average serving 60 more customers and the average delay per customer decreases. Thus, the proposed policy enables business growth while providing better customer service and offering drivers more opportunities.

7 Conclusion

In this paper, we introduce the restaurant meal delivery problem. The problem encompasses many stakeholders who have different objectives and who must be satisfied if restaurant meal delivery companies are to build successful businesses. In addition to satisfying the multiple stakeholders, the problem presents the challenge of being highly dynamic and uncertain.

To overcome the challenges, we propose an approach based on a cost-function approximation. The CFA relies on time buffers and postponement to account for the dynamism and uncertainty. 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. We also demonstrate that the policy offers the opportunity for growth while not diminishing customer satisfaction but while also 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 is easily extended to include restaurant- and customer-dependent deadlines.

Methodologically, our CFA currently uses a static time buffer. Future research could seek to identify state-dependent buffers. Alternatively, one might consider making decisions directly via the Bellman Equation by 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, one could discount delivery fees for customers who are willing to take delivery a half hour later than the regular deadline. Relatedly, restaurants who guarantee ready times could receive discounts on their fees. Determining when, which, and what size incentives are unexplored questions. 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.

References

Archetti, C., O. Jabali, and M. G. Speranza (2015). Multi-period vehicle routing problem with due dates. *Computers & Operations Research 61*, 122–134.

Arda, Y., Y. Crama, D. Kronus, T. Pironet, and P. Van Hentenryck (2014). Multi-period vehicle loading with stochastic release dates. *EURO Journal on Transportation and Logistics 3*(2), 93–119.

Azi, N., M. Gendreau, and J.-Y. Potvin (2012). A dynamic vehicle routing problem with multiple delivery routes. *Annals of Operations Research 199*(1), 103–112.

Berbeglia, G., J.-F. Cordeau, and G. Laporte (2010). Dynamic pickup and delivery problems. *European journal of operational research 202*(1), 8–15.

Boscoe, F. P., K. A. Henry, and M. S. Zdeb (2012). CA Nationwide Comparison of Driving Distance Versus Straight-Line Distance to Hospitals.

Cattaruzza, D., N. Absi, and D. Feillet (2016). The multi-trip vehicle routing problem with time windows and release dates. *Transportation Science 50*(2), 676–693.

Cortès, C. E., A. Núñez, and D. Sàez (2008). Hybrid adaptive predictive control for a dynamic pickup and delivery problem including traffic congestion. *International Journal of Adaptive Control and Signal Processing 22*(2), 103–123.

Cortès, C. E., D. Sàez, A. Núñez, and D. Muñoz Carpintero (2009). Hybrid adaptive predictive control for a dynamic pickup and delivery problem. *Transportation Science 43*(1), 27–42.

Fagerholt, K., B. A. Foss, and O. J. Horgen (2009). A decision support model for establishing an air taxi service: A case study. *The Journal of the Operational Research Society 60*(9), 1173–1182.

Franck, T. (2017, July 12). Home food delivery is surging thanks to ease of online ordering, new study shows. `https://www.cnbc.com/2017/07/12/home-food-delivery-is-surging-thanks-to-ease-of-online-ordering-new-study-shows.html`, accessed on September 21, 2017.

Fu, M. C. (2015). *Handbook of Simulation Optimization*. Springer.

Ghiani, G., E. Manni, A. Quaranta, and C. Triki (2009). Anticipatory algorithms for same-day courier dispatching. *Transportation Research Part E: Logistics and Transportation Review 45*(1), 96–106.

Ghiani, G., E. Manni, and A. Romano (2017, July). Scalable anticipatory policies for the dynamic and stochastic pickup and delivery problem. Submitted for publication.

Hyytiä, E., A. Penttinen, and R. Sulonen (2012). Non-myopic vehicle and route selection in dynamic darp with travel time and workload objectives. *Computers & Operations Research 39*(12), 3021–3030.

Isaac, M. (2016, February 11). Delivery Start-Ups Face Road Bumps in Quest to Capture Untapped Market. *New York Times*, B1.

Klapp, M. A., A. L. Erera, and A. Toriello (2016). The dynamic dispatch waves problem for same-day delivery. Available from `https://pdfs.semanticscholar.org/03eb/e7a9cc69ab53a00e7d385e5201a83421da86.pdf`, accessed on August 1, 2017.

Klapp, M. A., A. L. Erera, and A. Toriello (2017). The one-dimensional dynamic dispatch waves problem. *Transportation Science*, To Appear.

Macmillan, D. (2016, January 21). Uber Prepares to Launch Meal-Delivery Service in 10 Cities. *The Wall Street Journal*, B8.

Maze, J. (2016). Not everything delivers: Saying no to delivery. `http://nrn.com/operations/not-everything-delivers-saying-no-delivery/` (accessed July 12, 2016.

Mitrović-Minić, S., R. Krishnamurti, and G. Laporte (2004). Double-horizon based heuristics for the dynamic pickup and delivery problem with time windows. *Transportation Research Part B: Methodological 38*(8), 669–685.

Mitrović-Minić, S. and G. Laporte (2004). Waiting strategies for the dynamic pickup and delivery problem with time windows. *Transportation Research Part B: Methodological 38*(7), 635–655.

Muñoz Carpintero, D., D. Sàez, C. E. Corès, and A. Núñez (2015). A methodology based on evolutionary algorithms to solve a dynamic pickup and delivery problem under a hybrid predictive control approach. *Transportation Science 49*(2), 239–253.

Núñez, A., C. E. Cortès, D. Sàez, B. De Schutter, and M. Gendreau (2014). Multiobjective model predictive control for dynamic pickup and delivery problems. *Control Engineering Practice 32*, 73–86.

Powell, W. B. (2011). *Approximate Dynamic Programming: Solving the Curses of Dimensionality* (Second ed.). Wiley Series in Probability and Statistics. John Wiley & Sons, Inc.

Powell, W. B. (2017, July). A unified framework for stochastic optimization. Available from `http://castlelab.princeton.edu/Papers/Powell-UnifiedFrameworkStochasticOptimization_July222017.pdf`, accessed on August 1, 2017.

Psaraftis, H. N., M. Wen, and C. A. Kontovas (2016). Dynamic vehicle routing problems: Three decades and counting. *Networks 67*(1), 3–31.

Reyes, D., A. L. Erera, and M. W. Savelsbergh (to appear). Complexity of routing problems with release dates and deadlines. *European Journal of Operational Research*.

Rosenkrantz, D. J., R. E. Stearns, and P. M. Lewis (1974). Approximate algorithms for the traveling salesperson problem. In *Switching and Automata Theory, 1974., IEEE Conference Record of 15th Annual Symposium on*, pp. 33–42. IEEE.

Sàez, D., C. E. Cortès, and A. Núñez (2008). Hybrid adaptive predictive control for the multi-vehicle dynamic pick-up and delivery problem based on genetic algorithms and fuzzy clustering. *Computers & Operations Research 35*(11), 3412–3438.

Sayarshad, H. R. and J. Y. Chow (2015). A scalable non-myopic dynamic dial-a-ride and pricing problem. *Transportation Research Part B: Methodological 81*, 539–554.

Schilde, M., K. F. Doerner, and R. F. Hartl (2011). Metaheuristics for the dynamic stochastic dial-a-ride problem with expected return transports. *Computers & operations research 38*(12), 1719–1730.

Statista Survey (2016, November). How often do you order food for takeout? [Online; accessed 6-December-2016].

Ulmer, M. W. (2017, June). Dynamic pricing for same-day delivery routing. Available from `http://web.winforms.phil.tu-bs.de/paper/ulmer/Ulmer_pricing.pdf`, accessed on August 1, 2017.

Ulmer, M. W., J. C. Goodson, D. C. Mattfeld, and B. W. Thomas (2017, April). Route-based markov decision processes for dynamic vehicle routing problems. Available from `https://www.researchgate.net/publication/313421699_Route-Based_Markov_Decision_Processes_for_Dynamic_Vehicle_Routing_Problems`, accessed on August 2, 2017.

Ulmer, M. W., B. W. Thomas, and D. C. Mattfeld (2016, July). Preemptive depot returns for a dynamic same-day delivery problem. Available from `http://www.researchgate.net/publication/305433476_Preemptive_Depot_Returns_for_a_Dynamic_Same-Day_Delivery_Problem`, accessed on August 1, 2017.

U.S. Census Bureau, P. D. (2017, March). Annual estimates of the resident population: April 1, 2010 to July 1, 2016. `https://factfinder.census.gov/faces/tableservices/jsf/pages/productview.xhtml?src=bkmk`, accessed on September 28, 2017.

Voccia, S. A., A. M. Campbell, and B. W. Thomas (2017). The same-day delivery problem for online purchases. *Transportation Science*, To Appear.

Vonolfen, S. and M. Affenzeller (2016). Distribution of waiting time for dynamic pickup and delivery problems. *Annals of Operations Research 236*(2), 359–382.

Zeithaml, V. A., A. Parasuraman, and L. L. Berry (1990). *Delivering quality service: Balancing customer perceptions and expectations*. Simon and Schuster.

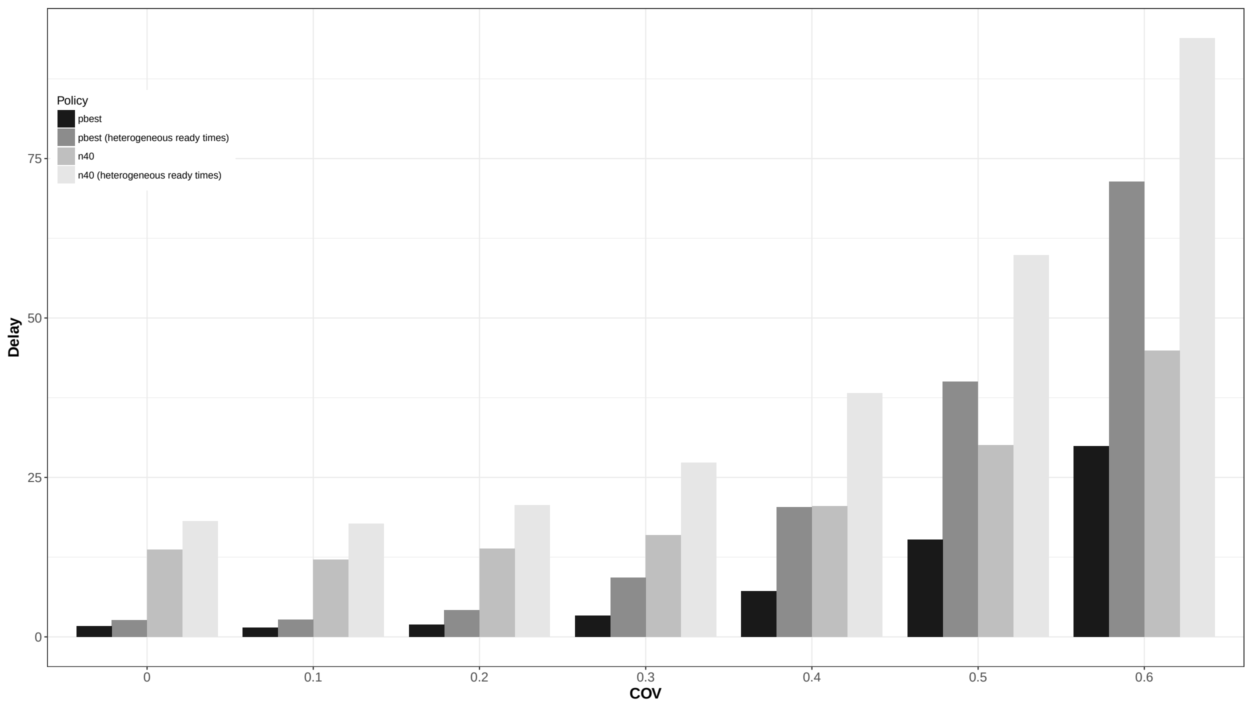


Figure 18: Solution Quality for Heterogeneous Ready Times

Appendix

A Heterogeneous Ready Times

In our computational experiments, we assumed a single type of restaurant with expected ready times of ten minutes. In practice, the expected ready times may vary for different types of restaurants. To show that our policy also performs well in this setting, we generate instance settings with 240 orders, CoV between 0.0 and 0.6, and three types of restaurants: fast (5 minutes expected ready times), fast casual (10 minutes), and gourmet (15 minutes). We assume 36 restaurants to be fast, 38 to be fast casual, and 36 to be gourmet restaurants. On average over all restaurants, the ready times are therefore still 10 minutes. For these instance settings, we conduct the same tuning procedure as for the conventional, homogeneous instance settings.

We compare the delay for the two policies $pbest$ and $n40$ in the heterogeneous setting with the values of the two policies in the homogeneous setting. The results for different CoVs are show in Figure 18. We observe that the tendency between policies $pbest$ and $n40$ remains the same also for the heterogeneous instances. But the values for both policies are significantly higher compared to the homogeneous instance settings. This can be explained by the restaurants with long expected ready times of 15 minutes and the CoV. For these restaurants, the time for delivery is already shorter

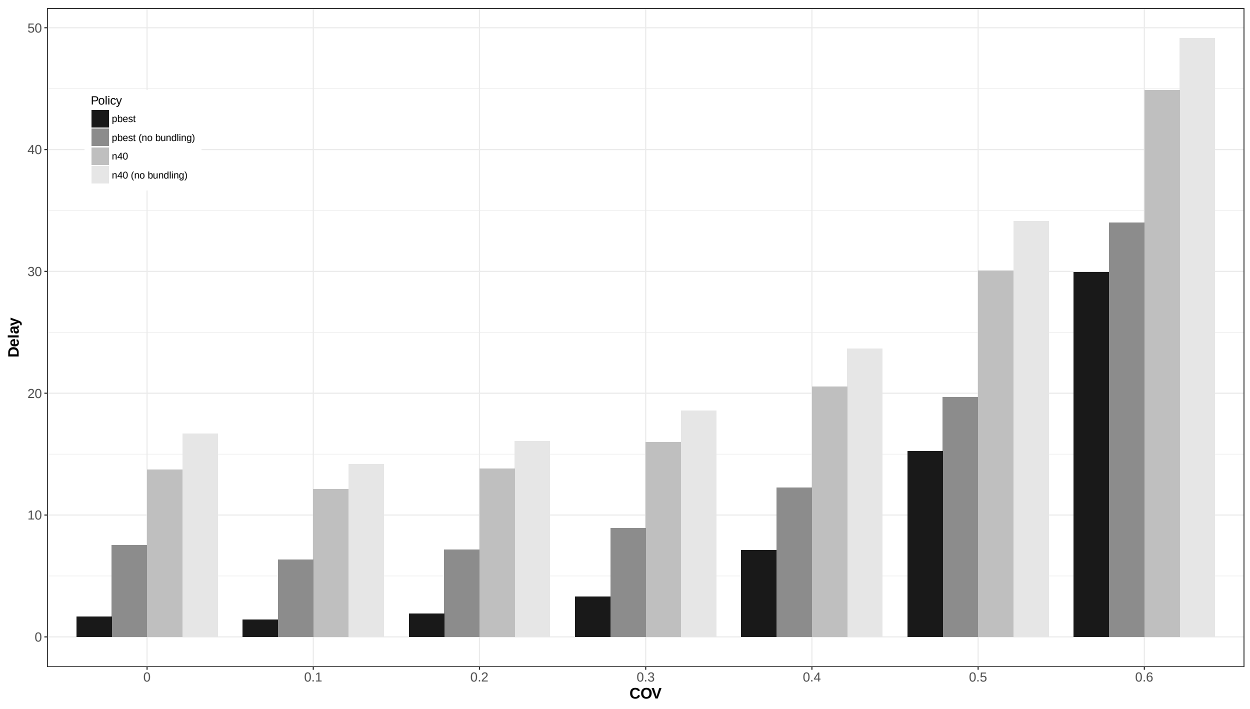


Figure 19: Impact of Bundling Options

with on average 25 minutes. Additionally, the CoV has a higher absolute impact of these ready times. From a managerial perspective,

B No Bundling

Finally, we analyze how bundling operations contribute to maintaining flexibility. To this end, we conduct experiments using the same policies as before, but prohibit bundling operations. We test the modified policies for the instance settings with 240 orders and CoVs between 0.0 and 0.6. We conduct the same tuning procedure as for the conventional policies. The results are shown in Figure 19. The x-axis shows the CoV, the y-axis shows the solution quality. We observe that without bundling both policies perform worse than with bundling. Still, the general tendency between the two policies $pbest$ and $n40$ is maintained. Our policy is therefore also applicable to business models where bundling is not an option, for example, in truckload routing or passenger transportation.