Market Thickness in Online Food Delivery Platforms: The Impact of Food Processing Times

Yanlu Zhao
Durham University Business School, yanlu.zhao@durham.ac.uk,

Felix Papier
ESSEC Business School, papier@essec.edu,

Chung-Piaw Teo
National University of Singapore, bizteocp@nus.edu.sg.

Problem definition: Online Food Delivery (OFD) platforms have witnessed rapid global expansion, partly driven by shifts in consumer behavior during the COVID-19 pandemic. These platforms enable customers to order food conveniently from a diverse array of restaurants through their mobile phones. A core functionality of these platforms is the algorithmic matching of drivers to food orders, which is the focus of our study as we aim to optimize this driver-order matching process.

Methodology/results: We formulate real-time matching algorithms that take into account uncertain food processing times to strategically 'delay' the assignment of drivers to orders. This intentional delay is designed to create a 'thicker' marketplace, increasing the availability of both drivers and orders. Our algorithms utilize machine learning techniques to predict food processing times, and the dispatching of drivers is subsequently determined by balancing costs for idle driver waiting and for late deliveries. In scenarios with a single order in isolation, we show that the optimal policy adopts a threshold structure. Building on this insight, we propose a new k-level thickening policy with driving time limits for the general case of multiple orders. This policy postpones the assignment of drivers until a maximum of k suitable matching options are available. We evaluate our policy using a simplified model and identify several analytical properties, including the quasi-convexity of total costs in relation to market thickness, indicating the optimality of an intermediate level of market thickness. Numerical experiments with real data from Meituan show that our policy can yield a 54% reduction in total costs compared to existing policies.

Managerial implications: Our study reveals that incorporating food processing times into the dispatch algorithm remarkably improves the efficacy of driver assignment. Our policy enables the platform to control two vital market parameters of real-time matching decisions: the number of drivers available to pick up and deliver an order promptly, and their proximity to the restaurant. Based on these two parameters, our algorithm matches drivers with orders in real-time, offering significant managerial implications.

Key words: Online food delivery, Market thickness, Dynamic matching, Bipartite min-cost matching
History:

1. Introduction

The online food delivery (OFD) market has been expanding at an annual rate of 25% over the past five years (Curry 2021). This growth has led to the rise of OFD platforms such as Meituan, UberEats, and GrabFood (Chen et al. 2020). The COVID-19 pandemic in 2020 further accelerated this trend as social distancing measures forced many restaurants to close their in-person dining services, leaving them reliant on takeaway and delivery options. Many restaurants worldwide were able to survive solely because of the OFD service (Chen and Hu 2020). Consequently, the top four OFD platforms in the U.S. – DoorDash, UberEats, Grubhub, and Postmates – experienced a revenue increase of USD 3 billion in the first half of 2020 (Levi Sumagaysay 2020).

OFD platforms operate in a highly competitive landscape where operating and marketing costs often exceed sustainable revenues. A significant contributor to these high operating costs is the strict service requirements, particularly the consumer expectation for reliable and quick deliveries, often within minutes of placing an order. According to a large-scale survey conducted by McKinsey (Hirschberg et al. 2016), more than 60% of customers indicated that timely deliveries and meeting delivery targets are crucial factors in OFD services. Failure to meet these targets can result in customers abandoning the platform, leading to a significant loss of revenue (Mao et al. 2019, Hasija et al. 2020). However, meeting delivery targets necessitates a high availability of drivers, thereby elevating operating costs. For example, Meituan's financial report revealed that the costs of their OFD business grew by 36% in 2019, primarily due to increased expenses related to delivery riders (Meituan 2020).

There are also various other operational challenges in addition to the financial pressures. For instance, the stock price of the OFD platform Deliveroo plummeted by as much as 31% within minutes of its London debut in March 2021. One contributing factor was the scrutiny the company faced regarding its treatment of food delivery drivers (Gemmell 2021). There is a growing body of literature that discusses how food delivery platforms exert control over their drivers, often more so than other types of delivery services. Today, millions of delivery drivers are effectively 'managed' by algorithms. They are continuously monitored through customer ratings and are influenced by various behavioral nudges, leading to new societal challenges (Sun 2019). These conditions have prompted drivers to evaluate whether they should disclose their genuine preferences and to develop strategies for selecting jobs, aiming to optimize their earnings (see, e.g., temporal pooling in Chen and Hu 2020).

The task of assigning orders to drivers is arguably the most crucial aspect of platform operations. Effective matching seeks to balance the costly idle time for drivers against the erosion of customer goodwill due to late deliveries. While the importance of effective matching extends across various sectors – from kidney exchanges and job searches to online dating (Roth 2015) – what sets OFD platforms apart are the immediate nature of the assignments, the significant costs and goodwill losses associated with delayed deliveries, and the substantial variability in food processing times. Our paper aims to contribute to this field by specifically examining the role of food processing time in the order assignment process. We aim to address the following research questions: (a) How can matching policies for OFD platforms effectively incorporate food processing times to delay matching decisions? (b) What impact does this delayed matching have on total costs and other performance metrics for the platform?

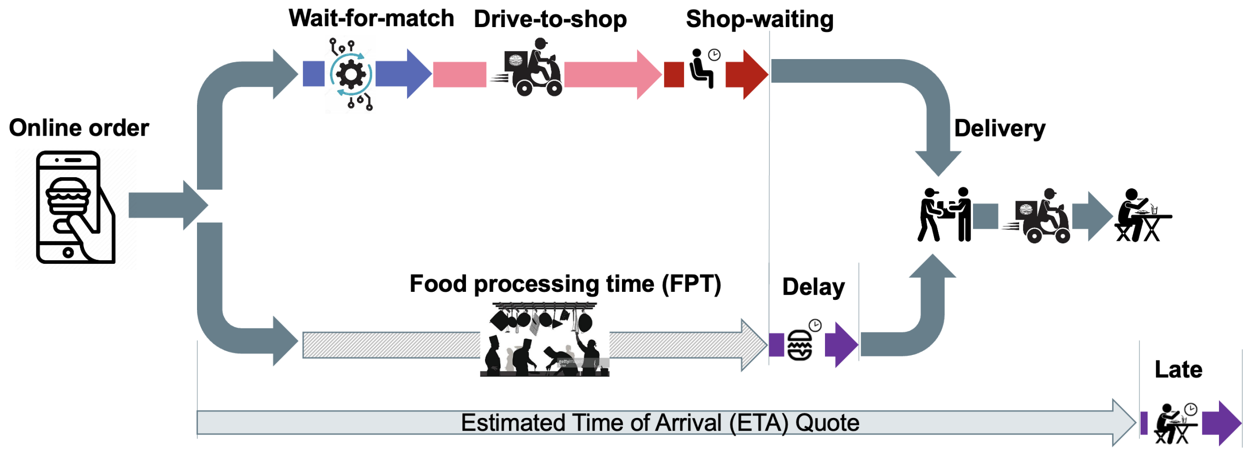


Figure 1 **Order matching and food dispatching process in online food delivery platform.**

OFD Process: Our research is motivated by the OFD operations (Figure 1) at Meituan, which involve two main activities: *order matching* and *food dispatching*. When a *customer* decides to place a meal order, the platform's web portal shows a selection of nearby restaurants. The customer then chooses her preferred dish from a specific restaurant and places the order. Upon receiving the order, the platform notifies the *provider*, such as a restaurant or shop, to prepare the food as requested. The *service time* for the platform is defined as the total duration from the moment an order is placed until the food is delivered to the customer. This duration is subject to a service time target set by the platform. A *driver* is assigned to pick up and deliver each order. Food can only be picked up from the restaurant when it is ready. In this study, we do not focus on the dispatching process after an order is picked up, as it does not directly influence the OFD matching problem. If a driver arrives at the restaurant before the order is ready, she must wait. Conversely, if she arrives after the food is ready , it results in a late delivery, which can lead to decreased food quality and lower customer satisfaction. Food can only be picked up from the restaurant when it is ready. In this study, we do not focus on the dispatching process after an order is picked up, as it does not directly influence the OFD matching problem. If a driver arrives at the restaurant before the order is ready, she must wait. Conversely, if she arrives after the food is ready (delayed pick up), it results in a late delivery, which can lead to decreased food quality and lower customer satisfaction. The platform must strike a balance between *driver waiting time* and the *lateness of deliveries* (Ryan et al. 2018). *Food Processing Time* (FPT) is defined as the time it takes for the restaurant to prepare the food once an order is received. This duration is typically unknown to the platform. Some platforms provide customers with a guaranteed *Estimated Time of Arrival* (ETA). In these cases, FPT in our model can be replaced by the time difference between the promised delivery time, offset by the travel time from the restaurant to the customer, and the time the request arrived at the system.

At Meituan, the average time to fulfill a food order ranges from 32 to 38 minutes. This duration includes the time spent waiting for an order to be matched, driving to the shop, waiting at the shop, and delivering the food to the customer. *Shop-waiting time* refers to the period between the driver's arrival at the restaurant and the moment the food is ready for pickup. Based on our data (Figure 2), shop-waiting time accounts for 35% of the total service time for an order, which is nearly as long as the time for delivery to the customer (41%). According to a study by Meituan (2021), delays in food preparation are a major cause of tension between drivers and restaurants, because these delays result in significant waiting time at shops. Similar issues related to shop-waiting time have been reported by Uber Eats and Deliveroo drivers (Ryan et al. 2018, Bosredon 2021). Feedback from drivers indicates that there are some flaws in the current delivery process, stating "The pickup is sometimes not ready at all... sometimes [it takes] even 20 minutes."(Ryan et al. 2018)

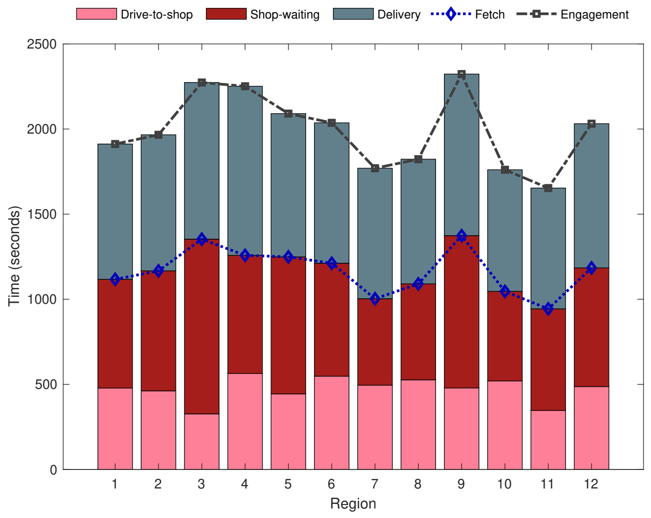


Figure 2 Processing times in different stages of OFD in different regions in which the company operates.

Most previous research has primarily focused on the question of *'Whom to match?'* in order to find the closest driver for an order. This is typically achieved with routing optimization and other techniques. However, few studies have addressed the issue of minimizing the shop-waiting time, a period for which the OFD platform must pay the driver. This shop-waiting time is as important as the drive-to-shop time. Ryan et al. (2018) developed a trip-inference model and a machine learning optimization tool specifically aimed at reducing the average time drivers spend waiting, walking, and parking at individual restaurants.

However, they did not propose a general framework for making system-wide matching decisions. Recently, Mao et al. (2022) analyzed an OFD platform's data set and found that drivers typically wait almost six minutes on average at restaurants. They asked the same question we examine in this paper: how can drivers be better matched to minimize time spent waiting for orders? Note that the opportunity costs of 'waiting for matching' and 'shop waiting' differ: In the former situation, drivers have the option to accept other jobs, while in the latter, they are unable to take on other work. Therefore, minimizing shop-waiting time is crucial for the platform.

Food processing begins as soon as an order is placed on the platform[1], even before driver matching takes place. In our data set, the FPT is not known in advance and is highly variable, with an average of 23 minutes and a coefficient of variation of 0.52. Despite this variability, advanced prediction techniques can forecast FPT using data from past orders. Several studies have already incorporated ETA values into research related to on-demand platforms (Liu et al. 2021b, Zehtabian et al. 2022, Cohen et al. 2022, Tao et al. 2022). In this paper, we develop matching algorithms designed to minimize total costs by taking into account both driving time and shop-waiting time. Specifically, we leverage the FPT to answer the question of *'When to match?'*. Our approach falls under the umbrella of real-time supply chain analytics, as we use extensive real-time operational data from various sources to enhance visibility and improve decision-making (Gupta 2022).

Specifically, we investigate whether intentionally delaying the matching of an order to a driver offers advantages in finding better matches. This concept, known as *market thickening*, was introduced by Akbarpour et al. (2020) in the context of online gaming and kidney exchanges. The FPT in OFD gives the platform the leeway to delay matching without compromising food quality (because the driver has to wait until the food being ready). However, excessive delays can result in higher costs due to late deliveries. The platform must therefore strike a careful balance between waiting to improve market thickness with better, shorter drive-to-shop matches and minimizing the costs of delivery delays. The complexity of this decision problem is compounded by the high variability of the FPT and the real-time influx of orders across the entire delivery area. Many OFD platforms use a batching policy to build market thickness, where orders are accumulated and drivers are assigned at regular intervals. In this paper, we demonstrate that our method outperforms the batching policy even when batch intervals are optimally chosen. The underlying algorithmic idea of our approach is straightforward: monitor the number of drivers who can fulfill an order within a specific range and assign the order only if the number of available drivers falls below a certain threshold, denoted as k. One of the challenges is to identify the optimal value for k. Our

[1] This is a conservative way to describe the food processing activities as it is challenging for platforms to track when meal preparation actually begins in the kitchen (Mao et al. 2022).

computational results indicate that performance is best when k is set at a moderate level - neither too high nor too low.

The order-driver matching problem discussed above can be categorized as a *Bipartite Min-cost Matching* (BMM) problem. We explore the properties of an optimal policy, set forth upper and lower bounds, and introduce two new policies: a k-level thickening (KT) policy for scenarios with known FPT and a buffered k-level thickening (BKT) policy for cases with uncertain FPT. Our paper makes three key contributions:

- To the best of our knowledge, this is the first study to investigate the impact of FPT in the OFD sector on the resulting matching cost and driver waiting time.
- We formulate the original BMM problem as a dynamic matching model. In the specific case of a single order in isolation, we analytically demonstrate that the optimal matching decisions adhere to a state-dependent threshold policy. Inspired by this threshold structure in the single-order scenario, we develop the driving-time-constrained KT policy to address the general problem.
- Through numerical experiments using real-world data, we show that our BKT policy decreases the total costs per order by 54% compared to existing real-case matching policies.

2. Literature Review

Our research is situated at the intersection of literature on dynamic matching, market thickness, and online food delivery. In this section, we outline the main contributions from each domain and elaborate on how we build upon them.

Research on dynamic matching: In recent years, there has been a surge in studies related to dynamic matching, particularly due to the emergence of the *on-demand economy* (Chen et al. 2020). Hu and Zhou (2022) investigated stochastic and dynamic matching control within a two-sided discrete-time system. Özkan and Ward (2020) introduced dynamic matching and pricing policies derived from a continuous linear program. To minimize unmatched requests, Banerjee et al. (2016) developed state-dependent matching policies that take into account spatial demand heterogeneity. Varma et al. (2022) explored a two-sided queuing system, focusing on joint pricing and matching controls.

Most of the aforementioned studies have considered matching policies in a myopic manner; that is, they instantly match demand with supply. However, this approach is unlikely to be optimal, as it might be more advantageous to reserve potential demand or supply for future periods. Akbarpour et al. (2020) explored this trade-off between matching frequency and market thickness, developing the so-called *greedy* and *patient* matching mechanisms. These mechanisms were calibrated by Liu et al. (2019) using a data set from DiDi. Ashlagi et al. (2018) demonstrated the superiority of batching policies with a data set of New York City yellow cabs, while Yan et al. (2020) from Uber developed a *dynamic waiting* strategy to enhance matching probability and supply efficiency in ride-hailing practices. Castro et al. (2021) analyzed a matching system featuring both flexible and specialized agents. Manshadi et al. (2022) examined matching on online volunteering platforms. Aouad and Saritac (2022) delved into the dynamic, stochastic matching problem with limited times, developing matching algorithms that provide worst-case performance guarantees. We draw inspiration from the idea of market thickening and extend it to a setting with heterogeneous agents and variable edge weights that change in real-time.

Research on market thickness: Market thickness is defined as the number of participants in a market (McLaren 2003). In matching markets, a long-standing discussion revolves around whether thickness positively or negatively affects the effectiveness of matching. Gan and Li (2016) demonstrated that the probabilities of matches in a thin market are significantly lower than those in a thick market. Akbarpour et al. (2020) analyzed the impact of search friction on market thickness in online markets. Li and Netessine (2020) discovered that a higher market thickness does not necessarily correlate with higher matching rates, attributing this to the presence of abundant search friction. Liu et al. (2019) indicated that the incremental value of market thickness, brought about by strategic waiting, might lead to fewer matches. Chen (2019) employed an $M/M/\infty$ queue to investigate mismatch performance. Arnosti et al. (2021) illustrated that straightforward operational interventions could alleviate congestion in matching markets effectively. Cui et al. (2022) examined the influence of market thickness on delivery efficiency. Xie et al. (2022) demonstrated that a little delay yields benefits in real-time online decision-making. They provided evidence that the difference between the proposed online policy with a delay and an offline hindsight policy diminishes at an exponential rate as the length of delay increases. We contribute to this ongoing debate by showing that the benefits derived from market thickening exhibit a quasiconvex pattern, suggesting the existence of an optimal thickening level that balances the positive and negative effects.

Research on online food delivery: Our work also contributes to the OFD literature, which has garnered significant attention in recent years (Reyes et al. 2018, Mao et al. 2019, Liu et al. 2021b, Chen et al. 2022). The OFD routing problem poses particular challenges in last-mile delivery due to the stringent service level targets. Although it bears similarities to the same-day delivery problem (Voccia et al. 2019), the need for quick-response decisions to adhere to short delivery times distinguishes it fundamentally from same-day delivery. Mao et al. (2022) conducted a comprehensive analysis of OFD operations using an industrial data set, and our current study aligns with the future directions mentioned in their work. Furthermore, Mao et al. (2019) empirically demonstrated that delivery performance is vital for an OFD platform to retain customers, and Liu et al. (2021b) proposed a data-driven framework to model the OFD problem and optimize assignment decisions. Ulmer et al. (2021) addressed a stochastic dynamic pickup and delivery problem, considering random FPT and deadlines. Chen and Hu (2020) compared temporal pooling and dedicated delivery in an on-demand delivery dispatching system. Distinctly, our work diverges from existing research on OFD routing by incorporating intentional delay to foster a thicker market, thereby enhancing compatibility between orders and drivers and increasing matching opportunities.

3. OFD Model Description

In this section, we introduce several key components to characterize the dynamic matching model in the OFD business. Orders and drivers arrive at the platform according to Poisson processes with rates λ_o and λ_d, respectively. We assume that $\lambda_o < \lambda_d$ and that drivers abandon the system at exponential rate λ_a for system stability (see also, e.g., Aouad and Saritac 2022).

Orders: An order is described by tuple $x_i = (\tau_i, l_i, \xi_i)$, where τ_i is the arrival time, l_i is the restaurant location, and ξ_i is the FPT. Throughout the paper, we use index i for orders. Restaurant locations l_i are independently distributed random variables with probability density function (pdf) $f_r(l_i)$. FPTs ξ_i are independently distributed random variables with pdf $f_\xi(\xi_i)$, which includes the duration of all activities between order placement and the earliest time it can be fetched. We assume that the FPT is independent of the matching policy and the state of the system (See Appendix 7.4.1 for an extension to a model in which the FPT is affected by the level of order congestion in the restaurant). We also assume that the system planner can only observe the realization of the FPT ξ_i at the end of the food processing. However, the planner predicts the FPT at order arrival (e.g., with advanced machine learning techniques as in Liu et al. (2021b); see Section 5 for details about FPT prediction). We denote the predicted value by $\hat{\xi}_i$, and assume that this prediction is unbiased, i.e., $\hat{\xi}_i = \mathbb{E}[\xi_i]$. Let the variance of the prediction error $\hat{\xi}_i - \xi_i$ denoted by σ_ξ^2. We refer to the special case of exact predictions ($\sigma_\xi^2 = 0$) as *known FPT*. Orders can only leave the platform when they are matched with a driver. Let $\mathcal{I}_t$ represent the set of orders that arrived until time t.

Drivers: A driver is described by tuple $y_j = (\tau_j, l_j, t_j^a)$ with arrival time τ_j, driver location l_j, and driver abandonment time $t_j^a > \tau_j$. Throughout the paper, we use index j for drivers. Driver locations are independently distributed random variables with pdf $f_d(l_j)$. Drivers leave the platform either when they are matched with an order or when they abandon. Abandonment times t_j^a are not known to the planner before the abandonment. We assume that a driver can only deliver a single order at a time; otherwise our problem would integrate a routing problem, which is not in the scope of this research. Let $\mathcal{J}_t$ denote the set of drivers that arrived and have not abandoned by time t.

Driving times: For picking up orders at the restaurant, let $d_{ij}(t)$ denote the (deterministic) driving time from the location l_j of driver j to the location l_i of the restaurant of order i, if the driver begins the drive at time t, i.e., we allow driving times to depend, for example, on time of day (our model can be extended to integrate real-time estimation of non-deterministic driving times in which $d_{ij}(t)$ is determined at the time of matching). We only make the assumptions that $t + d_{ij}(t) < t' + d_{ij}(t')$ for $t < t'$, i.e., that a driver cannot arrive earlier by waiting, and that $d_{ij}(t) = d_{ij}(t + \delta)$ for some δ and any i, j, t, i.e., that there is periodicity (of length δ, e.g., daily, weekly) in the driving times. For the driving time from restaurant to customer, we assume that it is independent of the matching decision, and we therefore treat it as a constant value, denoted by $\overline{\psi}$.

Order-Driver-Matching: Based on the state of the system, the planner has to decide at what time to match an order with a driver. For a given sample path, and policy Ω (i.e., the rule with which to make matches), let $\mathcal{M}_t^{\Omega}$ be the set of order-driver pairs *already matched* by time t:

$$\mathcal{M}_t^{\Omega} = \{(i,j,t_{ij}) : \text{order } i \text{ and driver } j \text{ matched at time } t_{ij}, i \in \mathcal{I}_t, j \in \mathcal{J}_t, \max(\tau_i, \tau_j) \le t_{ij} \le \min(t, t_j^a)\}. \quad (1)$$

We consider a pair (i,j) between an order i and a driver j to be *compatible* at time t if the assignment does not result in a delay in delivery, i.e., if $t + d_{ij}(t) \le \tau_i + \xi_i$. Otherwise, we refer to the edge (i,j) as *expired*. We refer to the time epoch t_{ij}^c that solves equation $t_{ij}^c + d_{ij}(t_{ij}^c) = \tau_i + \xi_i$, i.e., the exact time at which driver j has to drive to the restaurant to arrive at the end of the food processing to pick up order i, as the *edge expiration time*. Assumption $t + d_{ij}(t) < t' + d_{ij}(t')$ for $t < t'$, ensures that there is a unique edge expiration time per order-driver pair. Note that the exact edge expiration time is known in advance to the planner only in the case of known FPT.

Cost model: The cost of an order incurs at the time when the order is matched or at the end of the time horizon if it is not matched. It comprises two types of costs: First, the *fetching cost*, which reflects the time a driver spends on fetching the order, consisting of the drive-to-shop time, $d_{ij}(t_{ij})$, and the potential shop-waiting-time at the restaurant (given by the positive difference between the arrival time of the driver at the restaurant and the time of order completion by the restaurant): $(\tau_i + \xi_i - t_{ij} - d_{ij}(t_{ij}))^+$. Second, the potential *cost of lateness*, which represents the loss in customer satisfaction and which is proportional to the delivery delay, $(t_{ij} + d_{ij}(t_{ij}) - \tau_i - \xi_i)^+$. For a given matching policy Ω, we write the expected total costs per order, W^{Ω}, as

$$W^{\Omega} = \lim_{t\to\infty} \mathbb{E}\left\{ \frac{1}{|\mathcal{I}_t|} \sum_{(i,j,t_{ij})\in\mathcal{M}_t^{\Omega}} \left[d_{ij}(t_{ij}) + (\tau_i + \xi_i - t_{ij} - d_{ij}(t_{ij}))^+ + c \cdot (t_{ij} + d_{ij}(t_{ij}) - \tau_i - \xi_i)^+ \right] \right\}, \quad (2)$$

where c is the per-time-unit penalty for delay in fetching. Without loss of generality, we normalize the per-time-unit fetch cost to 1. The objective of the model is to minimize the sum of fetching cost and cost of lateness. We denote the optimal policy by Ω^*, i.e., the policy that achieves the *minimal* expected cost per order ($W^* = W^{\Omega^*}$).

In the above model with known FPT, order and driver arrivals and driver abandonments are Markovian, but the food processing times are not Markovian. Hence, the model belongs to the class of *Semi-Markov Decision Processes (SMDP)*, which we formally establish in Appendix 7.2.

In our analysis, we also report on performance indicators beyond total costs. These indicators include average *drive-to-shop* times (V_p^{Ω}), average *in-shop waiting* times (V_w^{Ω}), average *delay per order* (V_d^{Ω}), *service level* (V_l^Ω), average *number of drivers* available (V_n^Ω), average amount of times the drivers are *engaged* (V_e^Ω), average amount of time the drivers are *idle* waiting for jobs (V_q^Ω), as well as average *net income per driver* per time unit (V_s^Ω); see Appendix 7.1 for detailed definitions.

4. Policy Development and Analysis

In this section, we analyze the dynamic matching problem defined in Section 3. We commence by establishing a property of the optimal matching policy and illustrating the value of waiting in matching. Subsequently, we define three benchmark policies that are used in practice and literature, and present a lower bound on the total costs (Subsection 4.1). We then derive the optimal matching policy for the special case of a single order in isolation (Subsection 4.2) and, building on this result, develop a heuristic matching policy for multiple orders (Subsection 4.3).

In classical matching models, agents arrive randomly to a bipartite graph. Each edge has a fixed weight representing the matching cost, with the objective of minimizing the sum of the weights of all selected edges. A more generalized model is the dynamic matching problem with time windows, a topic rarely investigated in existing research; see, for example, Cao et al. (2020) or Aouad and Saritac (2022). In these models, the introduction of time windows significantly complicates the analysis. Our model is further complex, incorporating both time windows and variable edge weights that change over time as an edge expires. To facilitate our analysis, we initially examine the properties of the cost function, which we rely on in the remainder of our study. The following condition is necessary for the optimal matching time.

PROPOSITION 1. *In the case that the FPT is known at order arrival ($\sigma_\xi^2 = 0$), it is never optimal to match order i with a driver j at a time t if the edge is not yet expired (i.e., $t + d_{ij}(t) < \tau_i + \xi_i$).*

All proofs can be found in Appendix 7.3. The concept underlying Proposition 1 is straightforward: the decision-maker is invariably better off waiting until the edge expires than matching when the edge is not expired. Nonetheless, the selection of the edge is not *a priori* apparent, a point illustrated in settings such as the Uber Marketplace (`https://www.uber.com/us/en/marketplace/matching/`), and further demonstrated in the subsequent example for our context.

EXAMPLE 1. (**Edge selection and value of waiting**) Let us illustrate the potential benefits of waiting through a simple example. In Figure 3, we compare *myopic* order-driver matching (left-hand side of the figure) with *optimal* order-driver matching (right-hand side of the figure). The horizontal axis of each side represents the geographical location, denoted as l, while the vertical axis represents the travel time along the geographical line, denoted as t. At time $t = 0$, order 1 arrives to the system at $l_1 = 3$ with FPT $\xi_1 = 3$ time units. A second order 2 arrives at $t = 2$ at location $l_2 = 0$ with FPT $\xi_2 = 3$ time units. A driver A is available at $t = 0$ and $l_A = 1$, and a second driver B arrives at $t = 1$ and $l_B = 4$. Let us analyze the performance of the two policies:

- *Myopic matching:* As depicted on the left-hand side of Figure 3, if the planner matches the order 1 with driver A *myopically* at time $t=0$, then the planner must match the order 2 with driver B at time $t=2$. The first matching incurs a cost of 3 (two units for driving and one for waiting until the food is ready); the second matching incurs a cost of $4+c(4-3)=4+c$. The total costs per order are consequently $(3+4+c)/2=3.5+0.5\cdot c$.
- *Optimal matching:* As depicted on the right-hand side of Figure 3, if the planner waits until time $t=2$ to match the order 1 with driver B and until $t=4$ to match the order 2 with driver A, the first matching incurs a cost of 1, and the second matching incurs a cost of 1, leading to total costs per order as $(1+1)/2=1$.

The cost difference between the two policies is $2.5+0.5\cdot c$, reflecting the value of waiting and optimal edge selection. The challenge, therefore, lies in determining the optimal matching combinations.

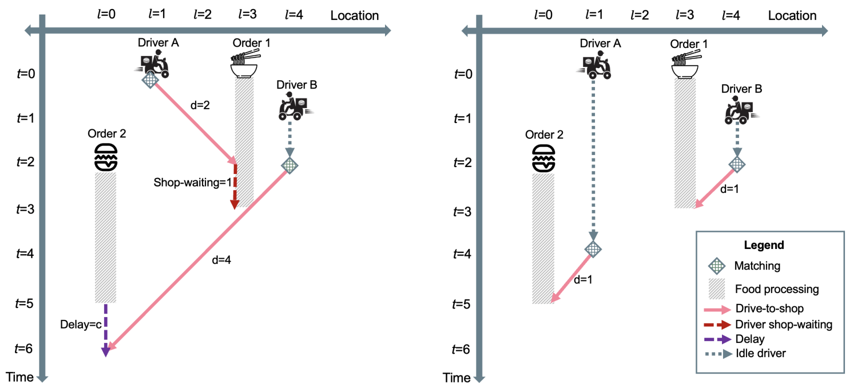


Figure 3 Example of myopic and optimal matching of two orders to two drivers.

4.1. Benchmark policies as lower and upper bounds

In this subsection, we introduce benchmark policies that act as lower and upper bounds for the performance of the optimal policy, as defined in the model presented in Section 3. Firstly, we describe the *Greedy policy*, denoted as Ω^g, which emulates the 'match-as-you-go' policies (Akbarpour et al. 2020) extensively employed in real OFD platforms, including platforms like Meituan.

Definition 1. (Greedy Policy Ω^g) *If an order i arrives at the platform at time t, then match it with the closest driver in $\mathcal{J}_t$ whenever $\mathcal{J}_t \neq \emptyset$. If a driver j arrives at the platform at time t, then match her with the closest order in $\mathcal{I}_t$ whenever $\mathcal{I}_t \neq \emptyset$.*

The immediate greedy matching policy exposes the benefits achievable through delaying the matching decision. It assigns drivers to potentially non-ideal orders, resulting in extended travel distances. Recognizing this, several OFD platforms are exploring alternative matching policies to enhance the efficiency of the dispatch process. For instance, Meituan, in a pilot project, has installed terminals (*ChuCanBao*) in restaurants, requesting chefs to update the FPT in real-time (Meituan 2021). Once the meal is ready, the planner greedily allocates the order to a specific driver. We summarize this policy as follows.

Definition 2. (Match-after-ready (MAR) Policy Ω^r) *When an order $i \in \mathcal{I}_t$ finishes processing at time t, then match it with the closest driver in $\mathcal{J}_t$ whenever $\mathcal{J}_t \neq \emptyset$. If a new driver j enters the market at time t, then match her with the closest, ready order in $\mathcal{I}_t$.*

Policy Ω^r emulates a *delayed* greedy policy, as it defers the myopic matching until the order is ready. Intuitively, this policy eliminates the drivers' waiting time at the restaurant. However, this improvement comes with a trade-off in food quality and customer satisfaction, as the prepared food must now await the arrival of the drivers.

We also evaluate the performance of the well-known *Batching* policy, denoted by Ω^b, that is employed by several online platforms (Yan et al. 2020). This policy groups jobs and drivers in intervals of ω time units (e.g., one minute) before executing a linear program to optimally match the available orders with drivers. Any unmatched orders and drivers are retained for the subsequent batch. In our simulations, we determine the optimal batching period ω through back-testing (see ONLINE Appendix 8.3 for a detailed formulation of the procedure).

Next, we define the optimal policy for a *full-information* case where FPTs, arrival and abandonment times, and real-time driver locations are known in advance. We determine the optimal policy $\Omega^{\mathcal{T}}$ over a finite horizon $\mathcal{T}$ by solving the following linear program.

$$\widetilde{W}^{\Omega^{\mathcal{T}}} = \min \frac{1}{|\mathcal{I}_{\mathcal{T}}|}\Bigg\{\sum_{i\in\mathcal{I}_{\mathcal{T}}}\sum_{j\in\mathcal{J}_{\mathcal{T}}}\sum_{t_{ij}\in\mathcal{T}} x_{ijt_{ij}}\left[d_{ij}(t_{ij}) + (\tau_i+\xi_i-t_{ij}-d_{ij}(t_{ij}))^+ + c\cdot(t_{ij}+d_{ij}(t_{ij})-\tau_i-\xi_i)^+\right]$$
$$+c\cdot\sum_{i\in\mathcal{I}_{\mathcal{T}}}(\mathcal{T}-\tau_i-\xi_i)\cdot(1-\sum_{j\in\mathcal{J}_{\mathcal{T}}}\sum_{t_{ij}\in\mathcal{T}} x_{ijt_{ij}})\Bigg\} \tag{3}$$

subject to

$$\sum_{j\in\mathcal{J}_{\mathcal{T}}}\sum_{t_{ij}\in\mathcal{T}} x_{ijt_{ij}} \le 1, \forall i\in\mathcal{I}_{\mathcal{T}};\ \sum_{i\in\mathcal{I}_{\mathcal{T}}}\sum_{t_{ij}\in\mathcal{T}} x_{ijt_{ij}} \le 1, \forall j\in\mathcal{J}_{\mathcal{T}} \tag{4}$$

$$x_{ijt_{ij}} \in \{0,1\}, \forall i\in\mathcal{I}_{\mathcal{T}}, t_{ij}\in\mathcal{T}, j\in\mathcal{J}_{\mathcal{T}};\ x_{ijt_{ij}}=0, \forall t_{ij}<\max(\tau_i,\tau_j) \text{ and } \forall t_{ij}>t_j^a. \tag{5}$$

The decision variables $x_{ijt_{ij}}$ indicate that an order i is matched with a driver j at time t_{ij}. Constraint (4) ensures each order is matched with at most one driver and vice versa. Constraint (5) ensures that only available orders and drivers are considered for matching. Since the scheduling horizon $\mathcal{T}$ is finite, we have to account for delay costs of orders that are not matched at the end of the horizon, which is represented by the last term in (3). We approximate the policy for the full-information case, denoted as Ω^f, by taking the limit of $\mathcal{T}$ as it approaches infinity, i.e., $W^{\Omega^f} = \lim_{\mathcal{T}\to\infty}\widetilde{W}^{\Omega^{\mathcal{T}}}$. It is evident that the benchmark policies and the full-information case provide upper and lower bounds to the optimal costs, a fact we establish in the following lemma (without proof).

Lemma 1. $\min\{W^{\Omega^g}, W^{\Omega^r}, W^{\Omega^b}\} \ge W^* \ge W^{\Omega^f}$.

4.2. Analysis of dynamic matching with a single order

In this section, we focus on a unique case of a single order as described in Section 3, without any competing orders for drivers. We aim to analytically determine the best strategy to understand the broader matching problem's core mechanisms.

We assume in this subsection that the FPT is known, i.e., $\sigma_\xi^2 = 0$, and that the driving time is constant and does not vary with the time of the day, i.e., $d_{ij}(t) \equiv d_{ij}$. We also assume that driver abandonment is minimal due to the short matching time frame, so that it has little or no effect on the matching decision. Without loss of generality, we set the order's arrival time to $\tau_i = 0$. We denote the *remaining* food processing duration at any given time t by ξ_i^r, which is given by $\xi_i^r = \tau_i + \xi_i - t$. For simplicity, we omit the order index i in our notation. For instance, d_j represents the driving time of driver j to the restaurant, instead of the full d_{ij}. Our analysis starts by identifying the optimal policy's potential matching times. Generally, if there is any active non-expired edge, it is preferable to wait until the last edge expires. This idea is captured in the following proposition:

Proposition 2. *For a single order scenario, it is never ideal to match a driver with the order while any edges remain unexpired.*

Considering the moment t_0 when the last edge expires (i.e., $d_j = \xi^r$), the decision-maker has two options:

- Instantly match driver j to the order, incurring a cost of d_j.
- Wait for a driver with shorter driving time.

If d_j is significant, waiting might be the better option. Our next theorem shows that this decision follows a threshold structure. The decision-maker will choose to match only if driver j's driving time is below a certain limit. This threshold depends only on the remaining FPT and not on prior driver arrivals.

Theorem 1. *In a single order scenario:*

(*a*1) *At the moment the last edge with driver j expires ($d_j = \xi^r$), the optimal decision is to match the order if and only if:*

$$d_j \leq \tilde{d}(\xi^r) = \tilde{d}(d_j), \tag{6}$$

where $\tilde{d}(\xi^r)$ is a threshold function dependent solely on the remaining FPT, ξ^r.

(*a*2) *If no drivers are currently available, upon the arrival of a new driver j with travel time and remaining FPT such that $d_j > \xi^r$, the order should be matched if and only if:*

$$d_j \leq \tilde{d}(\xi^r). \tag{7}$$

(*b*) *The threshold function, $\tilde{d}(\xi^r)$, is defined as:*

$$\tilde{d}(\xi^r) = \frac{W^+(\infty, \xi^r) + c\xi^r}{c+1}, \tag{8}$$

with $W^+(\infty, \xi^r)$ representing the expected future cost after the last non-expired edge's expiration when the remaining FPT is ξ^r. This function, $\tilde{d}(\xi^r)$, (non-strictly) increases with ξ^r.

Theorem 1(a) suggests that the optimal policy follows a threshold approach, wherein immediate matching is favorable only when the travel time of the last remaining driver falls below a certain threshold. Theorem 1(b) further establishes that for drivers arriving after their edge has expired, this threshold increases with the remaining FPT, ξ^r. Equivalently, the threshold decreases over time, indicating that the later the match occurs, the lower the likelihood of accepting a driver with a given travel time. This aligns with intuition: a driver arriving after the expiration of its edge is generally less preferred than one arriving prior to expiration.

To develop the aforementioned theorem, we demonstrate that, at any given moment, the driver possessing the shortest driving time to the order dominates all other drivers in the eligible driver pool. Consequently, the system exclusively relies on the driver with the minimum driving time. Subsequently, by employing the Markov property of the arrival stream, we ascertain that the net value of matching a particular driver at its edge expiration time dominates the net value of matching that same driver at any later moment. Therefore, if the net value of matching a driver is negative at the edge expiration time, it will always remain negative in the future, making it consistently advantageous to await the arrival of a more suitable driver. These two observations imply the threshold structure of the optimal policy, as established in Theorem 1.

Interestingly, an implication of Theorem 1 is that drivers satisfying the condition $d_j > \tilde{d}(\xi^r = d_j)$ will invariably be overlooked, as the system always opts to hold out for a more suitable match. However, fulfilling the condition $d_j \leq \tilde{d}(\xi^r = d_j)$ upon arrival does not ensure a match at the time of edge expiration, as a more close-by driver might arrive in the meantime.

The future expected cost, $W^+(\infty, \xi^r)$, ensuing after the expiration of the last edge, when no non-expired edges remain in the system (denoted by ∞ in the first argument), can be deduced from the function $W^+(d, \xi^r)$. This function represents the minimum future expected cost when the last edge to expire exhibits a driving time d and the remaining FPT is ξ^r. The function $W^+(d, \xi^r)$ always has a solution (see also the discussion in ONLINE APPENDIX 8.4) and can be evaluated numerically utilizing the following system of equations:

$$W^+(d, \xi^r) = \mathbb{E}_{d_{j'}} \mathbb{E}_{\hat{t}_{j'}} \begin{cases} \min\left\{d, W^+(\infty, \xi^r - d)\right\}, & \text{if } (\xi^r - d) \in \left[0, \hat{t}_{j'}\right], \\ W^+(\min\{d, d_{j'}\}, \xi^r - \hat{t}_{j'}), & \text{if } (\xi^r - d) > \hat{t}_{j'}, \\ \min\left\{d_{j'} + c\left[d_{j'} - \xi^r\right]^+, W^+(\infty, \xi^r - \hat{t}_{j'})\right\}, & \text{if } (\xi^r - d) < 0 \text{ and } d_{j'} > (\xi^r - \hat{t}_{j'}), \\ W^+(d_{j'}, \xi^r - \hat{t}_{j'}), & \text{if } (\xi^r - d) < 0 \text{ and } d_{j'} \leq (\xi^r - \hat{t}_{j'}). \end{cases} \tag{9}$$

and $W^+(d,\xi^r) = W^+(\infty,\xi^r)$ for any $d > \xi^r$, where j' is the next driver arrival with driving time $d_{j'}$ and inter-arrival time $\hat{t}_{j'}$.

We present an example to illustrate the application of the threshold policy. We selected a single restaurant and ten driver locations from the dataset introduced in Section 5. The locations were chosen to ensure that drivers have an equal probability of being in any of the ten locations, i.e., $f_d(l_j) \approx 1/10$ for $j = 1,...,10$. The average FPT is $\xi = 23$ minutes, and the arrival rate is $\lambda_d = 0.4$. The results are depicted in Figure 4. On the left-hand side, the blue dashed line represents the threshold $\tilde{d}(\xi^r)$ as a function of the remaining FPT ξ^r. If the driving time at edge expiration, d_j, (depicted by the red solid line) is below the threshold function (blue dashed line), the driver will be matched with the order. On the right-hand side of the figure, the driving radius around the restaurant corresponding to the threshold at varying values of ξ^r is displayed. Drivers with a driving time less than the time at which the blue dashed line and the red solid line on the left-hand side of Figure 4 intersect ($d_j = 13.02$ minutes), i.e., drivers located within the shaded circle area on the right-hand side, are potential candidates for matching at edge expiration. Those with a greater driving time, residing outside the shaded area, will never be matched. For instance, drivers with $d_j = 1$ arriving before $\xi^r = 1$ (i.e., before the edge expires) could be matched as $d_j = 1 < 10 = \tilde{d}(1)$; this also applies to drivers with $d_j = 11$. Conversely, drivers with $d_j = 17$ or $d_j = 20$ are excluded from matching. This observation elucidates that for drivers arriving before the edge expiration, a *single threshold value* determines whether the last non-expired driver is matched upon edge expiration.

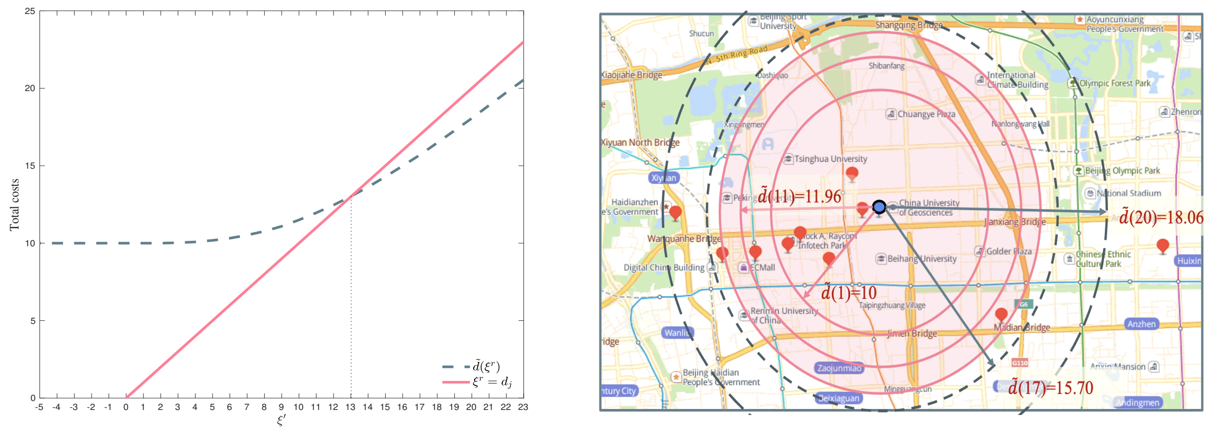


Figure 4 **Threshold function $\tilde{d}(\xi^r)$ as a function of remaining FPT ξ^r (left) and 'threshold' circles around the restaurant location for different values of ξ^r (right).**

In most real-world scenarios, orders do not occur in isolation, and drivers have the potential to be matched with other orders. However, the fundamental decision problem remains consistent in the general model: the immediate cost of matching a driver to an order, dependent on the driving time, must be weighed against the expected future cost of waiting. In the general model, the future cost of waiting likely depends not only on the remaining FPT but also on additional factors such as the presence of other outstanding drivers and orders. This renders the ‘threshold’ for acceptance versus waiting more complex than in the scenario of an isolated, single order. We leverage this insight in the following subsection to develop a matching policy for the general case, where a driving-time threshold is complemented by a thickening level to regulate competition between orders for the best drivers.

4.3. KT Policy Development and Analysis

In this section, we develop the KT policy for the model in Section 3, based on the insights that we have gained from our prior analysis. We describe the policy, analyze its properties, and develop a simple, closed-form approximation of its performance.

KT Policy with Known FPT. Compared to the single-order case, in the presence of multiple orders, it is not always optimal to wait until the last edge expires. This is because the corresponding driver might not be available anymore; it may have already been matched with another order. This risk of drivers ‘dropping out’ from the pool of eligible drivers for a given order renders it analytically infeasible to establish a result equivalent to Theorem 1 for the general case. We, therefore, develop a heuristic policy for the general case, incorporating a threshold structure with additional mechanisms to match earlier when necessary to mitigate the risk of drop-outs. We refer to this policy as the *driving-time-constrained k-level Thickening (KT) Policy.*

The ‘design’ of the KT policy comprises three ’mechanisms’, each grounded in our prior analysis. First, the policy constrains the neighborhood through a threshold function on the driving time, denoted by $\breve{d}(\xi^r)$, which is optimal in the single-order scenario (Theorem 1). Second, in contrast to the optimal strategy in the single-order case of awaiting the expiration of the last driver (see Proposition 2), the KT policy regulates the level of *market thickness* (refer to the discussion in Section 1) through a threshold parameter k. This parameter guards orders against the risk of a driver, for which the order is waiting to ‘drop out’, that is, being matched with a different order. A larger thickening level k results in reduced waiting times, albeit potentially leading to less desirable driving times; a smaller k extends the waiting period but could yield shorter driving times. Third, the KT policy also matches orders to the nearest available driver when *all* compatible drivers have dropped out, preventing unnecessary delays due to extended waiting times for incoming drivers, especially in situations with low driver arrival rate.

Before introducing the policy, let us define the *driving-time constrained neighborhood* $\mathcal{N}_t(i,d)$ of an order i at time t as the set of *compatible* drivers that can reach the restaurant of order i within driving time d, i.e., available drivers j for which $d_{ij}(t) \leq d$.

Definition 3. (*Driving-time-constrained KT Policy with known FPT,* $\Omega^{k,\breve{d}(\cdot)}$) *Let the parameter k represent the level of market thickness and $\breve{d}(\cdot)$ a threshold function. In the case of multiple orders,*

$(a1)$ *If, at the edge expiration time t^c_{ij} for an order i and driver j, driver j is within the driving-time constrained range of order i, i.e., $d_{ij}(t^c_{ij}) \le \breve{d}(d_{ij}(t^c_{ij})) = \breve{d}(\xi^r_i)$, and if the number of drivers in this neighborhood is less than or equal to k, i.e., $|\mathcal{N}_{t^c_{ij}}(i, \breve{d}(\xi^r_i))| \le k$, then order i is immediately matched with driver j. Moreover, for any other order $i' \neq i$ for which driver j was the last non-expired driver in its neighborhood, i.e., $|\mathcal{N}_{(t^c_{ij})^+}(i', \breve{d}(\xi^r_{i'}))| = 0$, where $(t^c_{ij})^+$ is the time epoch immediately following the matching (i,j), order i' is immediately matched with the nearest available driver, i.e., driver $\arg\min_{j'} d_{i'j'}(t^c_{ij})$.*

$(a2)$ *If, upon the arrival of a driver j, the edge expiration time t^c_{ij} with an order i has already passed, i.e., $t^c_{ij} < \tau_j$, and if driver j is within the driving-time-constrained range of order i, i.e., $d_{ij}(\tau_j) \le \breve{d}(\xi^r_i)$, and the neighborhood of this order is empty, i.e., $|\mathcal{N}_{\tau_j}(i, \breve{d}(\xi^r_i))| = 0$, then order i is immediately matched with driver j. If multiple orders fulfill these conditions, the order with the greatest lateness cost is selected for matching, i.e., the order i' where $i' = \arg\max_i (\tau_j + d_{ij}(\tau_j) - \tau_i - \xi_i)^+$.*

Statement (a1) of the KT policy prescribes an action for order-driver pairs that reach their edge expiration time. If the number of compatible drivers of the same order in the neighborhood falls below the threshold, the pair is matched; otherwise, it is not. Statement (a1) also concerns a second action: if the expiring pair is matched and if the matching leaves another order without any compatible drivers in the neighborhood, it is matched with the closest driver. This additional action avoids situations in which orders are 'starved' and have to wait for the arrival of new drivers. The statement (a2) of the KT policy is similar to the statement (a1) but concerns order-driver-pairs that are already expired at driver arrival. In such a situation, the action is similar to the one of (a1), i.e., the pair is also matched if the number of compatible drivers to the order is below the threshold.

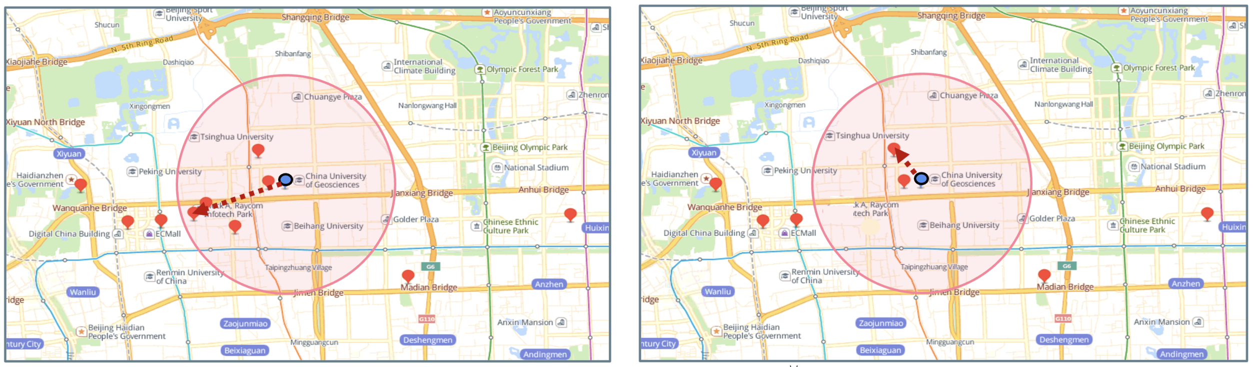


Figure 5 Illustration of the KT policy with two examples ($k = 2$ and $\breve{d}(\cdot) = 10$).

Figure 13 provides a simple example of the functioning of the KT policy with parameters $k = 2$ and $\breve{d}(\cdot) = 10$ based on two decision epochs, where the earlier point in time is on the left-hand side and the later point in time on the right-hand side. The blue dot in the center represents the order's restaurant location, while red dots represent idle drivers. The red circle represents the neighborhood defined by the policy parameter $\breve{d}(\cdot) = 10$. The dashed arrow indicates the edge between the order and a driver that is about to expire. In the figure on the left-hand side, five idle drivers are located within the neighborhood. Because $k = 2 < 5$, the driver of the expiring edge is not matched with the order. However, as time progresses, we come to the time epoch that is illustrated in the figure on the right-hand side. At this time, only two idle drivers are located in the neighborhood, which equals parameter $k = 2$ such that we immediately match the driver of the expiring edge with the order.

For subsequent analysis, let k^* denote the optimal thickening level. It is evident that the KT policy with $k = 1$ aligns with the well-known *Patient* policy proposed by Akbarpour et al. (2020). Moreover, the KT policy, with a sufficiently large value of k, is tantamount to matching when the first edge with an eligible driver expires. In the presence of a single order, the KT policy converges to the optimal policy, which we formally show in the following corollary.

COROLLARY 1. *In the case of a single order (see Section 4.2), the KT policy* $\Omega^{k,\breve{d}(\cdot)}$ *with* $k = 1$ *and* $\breve{d}(\cdot) \equiv \tilde{d}(\cdot)$ *coincides with the optimal policy.*

The result of Corollary 1 holds, because for $k = 1$ the policy always waits to match until the last edge expires, a strategy that is optimal in the single-order case (Proposition 2). Furthermore, the threshold policy, characterized by threshold function $\tilde{d}(\xi^r)$, results in optimal matching, as shown in Theorem 1. In Appendix 7.5, apply the KT policy to Example 1 to illustrate how the KT policy can coincide with the optimal policy for some cases.

KT Policy with Unknown FPT. In the case of unknown FPT ($\sigma_\xi > 0$), the FPT value manifests as a random variable, only revealed to the decision-maker once the food is prepared. Consequently, upon the arrival of an order, the decision-maker can only determine an edge expiration time based on the *predicted* FPT, $\hat{\xi}_i$. This scenario introduces the possibility that an order may expire earlier or later than anticipated. To address this, we extend our KT policy to the *Buffered KT (BKT) Policy*, which protects against FPT prediction errors by adding a buffer time $b^{\hat{P}}_{\sigma_\xi}$ to the predicted edge expiration time. The buffer $b^{\hat{P}}_{\sigma_\xi}$ is set such that the probability $P[\xi_i \le \hat{\xi}_i + b^{\hat{P}}_{\sigma_\xi}] = \hat{P}$ is satisfied, where policy parameter $\hat{P}$ is the *protection level*. Should the prediction error adhere to a Normal distribution, the buffer is assigned as $b^{\hat{P}}_{\sigma_\xi} = z_{\hat{P}} \cdot \sigma_\xi$, with $z_{\hat{P}}$ representing the $z-$score corresponding to $\hat{P}$.

We define the *'predicted', 'protected' expiration time* of an edge (i, j) with unknown FPT, denoted as t^{bc}_{ij}, to be the value satisfying the equation $t^{bc}_{ij} + d_{ij}(t^{bc}_{ij}) = \tau_i + \hat{\xi}_i + b^{\hat{P}}_{\sigma_\xi}$. This value is known to the planner upon the arrival of the order. However, it is possible that food preparation finishes even before the time t^{bc}_{ij}. Consequently, under the BKT policy, decisions have to be made at the earlier of the two events, referred to as the 'critical decision' time, given by $\min(t^{bc}_{ij}, \tau_i + \xi_i)$, a value unknown to the planner until either of the two events occurs.

Definition 4. (*BKT Policy with unknown FPT,* $\Omega^{k,\breve{d}(\cdot),\hat{P}}$) *In the case of multiple orders,*

$(a1)$ *If, at the 'critical decision' time* $\min(t_{ij}^{bc}, \tau_i + \xi_i)$ *of an order* i *and driver* j*, driver* j *is within the driving-time constrained range of order* i*, i.e.,* $d_{ij}(t_{ij}^{bc}) \le \breve{d}(d_{ij}(t_{ij}^{bc}))$*, and if* k *or fewer drivers are in the neighborhood, i.e.,* $|\mathcal{N}_{t_{ij}^{bc}}(i, \breve{d}(d_{ij}(t_{ij}^{bc})))| \le k$*, then order* i *is immediately matched with driver* j*. Furthermore, upon such a matching, for any other order* $i' \neq i$ *for which driver* j *was the last non-expired driver in the neighborhood of order* i'*, i.e.,* $|\mathcal{N}_{(t_{ij}^{bc})^+}(i', \breve{d}(\tau_{i'} + \hat{\xi}_{i'} + b_{\sigma_\xi}^{\hat{P}} - t_{ij}^{bc}))| = 0$*, where* $(t_{ij}^{bc})^+$ *is the time epoch right after the matching* (i,j)*, order* i' *is immediately matched with the nearest driver, i.e., driver* $\arg\min_{j'} d_{i'j'}(t_{ij}^{bc})$*.*

$(a2)$ *If, upon the arrival of a driver* j*, the 'critical decision' time* $\min(t_{ij}^{bc}, \tau_i + \xi_i)$ *with an order* i *has already passed, i.e.,* $\min(t_{ij}^{bc}, \tau_i + \xi_i) < \tau_j$*, and driver* j *is within the driving-time-constrained range of order* i*, i.e.,*

$$d_{ij}(\tau_j) \le \begin{cases} \breve{d}(\tau_i + \hat{\xi}_i + b_{\sigma_\xi}^{\hat{P}} - \tau_j), & \tau_j \le \tau_i + \xi_i, \\ \breve{d}(\tau_i + \xi_i - \tau_j), & \tau_j > \tau_i + \xi_i, \end{cases} \tag{10}$$

and if this neighborhood is empty, i.e.,

$$\begin{cases} |\mathcal{N}_{\tau_j}(i, \breve{d}(\tau_i + \hat{\xi}_i + b_{\sigma_\xi}^{\hat{P}} - \tau_j)| = 0, & \tau_j \le \tau_i + \xi_i, \\ |\mathcal{N}_{\tau_j}(i, \breve{d}(\tau_i + \xi_i - \tau_j)| = 0, & \tau_j > \tau_i + \xi_i, \end{cases} \tag{11}$$

then order i *is immediately matched with driver* j*. If multiple orders fulfill these conditions, the order with the greatest 'predicted' lateness cost is selected for matching, i.e., the order* i' *with*

$$i' = \arg\max_i \begin{cases} (\tau_j + d_{ij}(\tau_j) - \tau_i - \hat{\xi}_i - b_{\sigma_\xi}^{\hat{P}})^+, & \tau_j \le \tau_i + \xi_i, \\ (\tau_j + d_{ij}(\tau_j) - \tau_i - \xi_i)^+, & \tau_j > \tau_i + \xi_i. \end{cases} \tag{12}$$

In Appendix 7.4, we show how to adapt the BKT policy to different model extensions.

Simplified Performance Analysis of the KT Policy. Next, we explore the sensitivity of the performance of the KT policy with known FPT with respect to the level of market thickness k. As the original system is intractable, we perform this investigation on a simpler model, for which we can derive a closed form expression for the total costs. For this simplified model, we assume that:

- The driving time to the restaurant and FPT are constant values, denoted by $\overline{d}$ and $\overline{\xi}$, respectively. $\overline{d}$ represents the *non-compressible* driving time, i.e., the average driving time under maximum waiting (i.e., with $k = 1$); $\overline{\xi}$ represents the average FPT.
- The number of compatible drivers for an order at its arrival is a constant value denoted by $\tilde{k}$ (see also Banerjee et al. 2018, who use a similar assumption).[2]

[2] Using a fixed number of drivers for the analysis provides a more accurate approximation of the KT policy than a $M/G/\infty$ queue, in which servers represent drivers. An analysis of a queue-based approximation and a comparison of the accuracy of the two approximation models can be obtained from the authors upon request.

- The times between edge expiration events follow an exponential distribution with rate θ, and we assume drop-outs and new arrivals of drivers to be balanced during the time an order waits to be matched. As a consequence, the matching duration $t_{ij} - \tau_i$ (i.e., the time between arrival of an order and its matching) corresponds to the sum of $\tilde{k} - k + 1$ exponentially distributed delays with rates $\mu_j = j\theta$. We denote this random variable by $X(k)$. It follows a hypoexponential distribution with mean $\mu(k) = \sum_{j=k}^{\tilde{k}} \mu_j^{-1}$ and pdf $\tilde{f}_k(x)$.[3]

The values $\overline{d}$, $\tilde{k}$, and θ can be derived from the neighborhood definition, i.e., from the threshold function $\breve{d}(\cdot)$ of the KT policy. Based on the above simplifications, we obtain from Equation (2) the following expression for the total matching cost in the platform:

Proposition 3. *Expected total costs per order in the simplified system are given by*

$$\begin{aligned}\widetilde{W}^{\Omega^{k,\breve{d}(\cdot)}}(k) &= \overline{d} + \mathbb{E}[(\bar{\xi} - \overline{d}) - X(k)]^+ + c \cdot \mathbb{E}[X(k) - (\bar{\xi} - \overline{d})]^+ \\ &= \overline{d} + \int_0^{\bar{\xi}-\overline{d}} (\bar{\xi} - x - \overline{d})\tilde{f}_k(x)dx + c \cdot \int_{\bar{\xi}-\overline{d}}^{\infty} (x + \overline{d} - \bar{\xi})\tilde{f}_k(x)dx \\ &= \bar{\xi} - \mu(k) + (1 + c) \cdot p(k), \end{aligned} \tag{13}$$

with

$$p(k) = \mathbb{E}[X(k) - (\bar{\xi} - \overline{d})]^+ = \frac{e^{-k\theta(\bar{\xi}-\overline{d})}}{k\theta} \cdot \binom{\tilde{k}}{k} \cdot \sum_{m=0}^{\tilde{k}-k} (-1)^m \cdot \frac{\binom{\tilde{k}-k}{m}}{e^{m\theta(\bar{\xi}-\overline{d})}\left(1 + \frac{m}{k}\right)^2}. \tag{14}$$

The delay penalty is represented by the term $c \cdot p(k)$, where $p(k)$ represents the expected delay. The fetch cost is given by $\bar{\xi} - \mu(k) + p(k)$, representing the *effective* driving time under thickening level k as the sum of the non-compressible part of the driving time (for $k = 1$) plus the additional driving time from 'earlier' matching (for $k > 1$).

Proposition 3 provides insights into the structure of the total costs of the thickening policy. We can interpret $\mu(k)$ as the average duration that orders remain on the platform before they are matched. Since $\mu(k)$ is decreasing in k, the smaller the k, the longer the order stays in the system to allow the market to thicken. Fetch costs are intuitively decreasing with waiting (i.e., the smaller the k), while the penalty cost is intuitively increasing with the matching delay (i.e., with smaller k), which is evident from the structure of the term for expected delay, $\mathbb{E}[X(k) - (\bar{\xi} - \overline{d})]^+$. As fetch costs and penalty costs are influenced by the thickening level in opposite directions, the optimal thickening level k^* is likely the one that balances both types of costs. We formally establish that the expected cost per order is quasi-convex in the thickening level k and submodular in k and c in the following proposition.

[3] This assumption is reasonable, durations are frequently approximated by exponential distributions. In ONLINE Appendix 8.1, we show that the edge expiration times of our data set are well represented by the exponential distribution. Rate θ is determined by average driving times, arrival rates of orders and drivers, and other factors.

PROPOSITION 4. *For given $(\overline{d}, \tilde{k}, \theta)$, total costs per order in the simplified system are $\widetilde{W}^{\Omega^{k,\check{d}(\cdot)}}$, is*
(a) Quasi-convex in the thickening level k.
(b) Submodular in parameters k and c, such that $k^*(c)$ is (non-strictly) increasing in c.

Part (a) of Proposition 4 implies that the total costs initially decrease and then increase with the thickening level k, illustrating the fundamental trade-off of thickening in matching markets (see, e.g., Chen 2019). This result facilitates a straightforward search for the optimal k^*. The value derived from our simplified model can serve as a starting point in the search for the optimal policy parameters, a process we undertake in Section 5. Part (b) of Proposition 4 indicates that the parameters k and c are strategically complementary in the marginal cost (Topkis 1978). The net cost for an additional k does not increase with an increase in c, and vice versa. Consequently, the optimal thickening level $k^*(c)$ is monotone in the penalty unit cost c. The Greedy policy (Subsection 4.1) incurs total costs of $\overline{\xi}$ under the assumptions of our simplified model, meaning that the performance difference between the Greedy policy and the KT policy can be conveniently expressed as $W^{\Omega^g} - \widetilde{W}^{\Omega^{k,\check{d}(\cdot)}} = \mu(k) - (1+c)p(k)$. We observe that, for some values of k, the Greedy policy may outperform the KT policy; however, since the term $p(k)$ approaches zero for large values of k, a sufficiently large level of k can always be chosen such that the KT policy dominates the Greedy policy.

5. Numerical Experiments

In this section, we use the data set from the OFD platform *Meituan* in China to test the performance of different policies. After a description of the data set, we analyze the performance of the different policies under the assumption that the FPT is known. Then, we develop a prediction model for the FPT and apply the prediction model and our policies to the case in which the FPT is uncertain upon order arrival. More simulation details are provided in ONLINE Appendix 8.2.

5.1. Data Description and Preliminary Analysis

Meituan is the largest OFD platform in China. We have access to one month of data, from 3-30 July 2017. This data set includes more than 1 million detailed orders, drivers, and weather information collected from eleven regions with an area of $\sim 60km^2$ per region. Order information contains, among others, order ID, order arrival time, region, food provider ID, food value, locations of the food provider and the customer, arrival time of driver at the shop, and delivery begin and end time. The orders cover 2 447 food providers and 153 096 customers. Driver information contains minute-wise numbers of waiting and busy drivers in the network.

In our data set, the platform experiences an average number of 3 200 orders per day per region. Demand, however, varies considerably across time periods and in its spatial density. The temporal analysis of arrival times shows that nearly 40% of orders are made for lunch time (10:00-14:00) and 30% for dinner time (18:00-20:00). To simplify our experiments, we define three scenarios based on the arrival time of the order: *no-rush*, *rush*, and *overall*. The *no-rush* scenario is comprised of all orders that arrive in the period of 7:00-10:00, the *rush* scenario comprises all orders that arrive during the period 10:00-14:00, and the *overall* scenario includes all orders during a day (7:00-23:00). The *overall* order arrival rate is $\lambda_o = 3$ per minute. The average FPT values of the three scenarios in the data set are 14'28 (14 minutes and 28 seconds) for the no-rush-scenario, 25'42 for the rush-scenario, and 23'13 for the overall scenario. The standard deviation in the overall scenario is 11'94 minutes. More than 70% of FPT values are less than or equal to 30 minutes. We provide a detailed description of the data set in ONLINE Appendix 8.1.

The nonuniformity of the demand over the time of the day implies, however, that the demand for driver capacity also varies significantly during the day. Relying only on a fixed number of inhouse drivers would render it difficult to maintain a consistent service level during rush hours. Therefore, Meituan uses crowdsourced, on-demand drivers who provide the required flexibility to satisfy demand variation over the course of the day. Inhouse drivers are paid a fixed salary (3000 yuan per month) plus compensation for each delivered order. Crowdsourced drivers, on the other hand, only receive compensation for each delivered order, which is higher than the variable compensation of inhouse drivers. For ease of exposition, we assume that the unit fetch cost paid to both types of drivers are standardized to one (∼20 yuan/hour). See Appendix 7.4.2 for an extension in which drivers have type-specific compensation. We did not observe driver abandonment during the day in our data-set, and therefore use a negligible abandonment rate ($\lambda_a \to 0$). However, the impact of λ_a is further tested and presented in Appendix 7.4.6.

In our simulation, to characterize the one-to-one delivery process of our model, we assume that a driver has an average speed of $v = 18km/h$ and becomes available instantly after she finishes a delivery. We also assume that the previous matching neither affects the driver location after delivery nor the restaurant location of new demands. This assumption is reasonable, because when drivers are on time, the driving time from restaurant to customer is fixed. It allows us to mimic the driver arrivals in the data set since drivers appear as new arrivals upon service completion.

To determine the unit penalty cost c, the platform's marketing department has to assess the deterioration in customer experience per time unit of delay and the resulting loss in future revenues. In our numerical experiments, we use a base value of $c = 6$, which is six times the cost of a driver, i.e., $20 \times 6 = 120$ yuan per hour of delay; a value similar to the approx. USD 10-20 per order used in prior research (see Mao et al. 2019).

For each analysis, we numerically determine the optimal policy parameters (k^* and $\check{d}^*(\cdot)$ for the KT policy, and $\hat{P}^*$ with the other parameters for the BKT policy). We use a constant threshold value, $\check{d}$, instead of a threshold function, because estimating a function for each restaurant would be too complex.

We use the simplified model (see Subsection 4.3) for an initial calibration of the search for the policy parameters. This leads to substantial run time savings in the implementation of our policy in practice. After the first calibration, we can vary each parameter successively and compute the total costs with simulation, until a local optimum is found.

5.2. Performance Analysis with Known Food Processing Time

In this subsection, we assume that the exact FPT is known to the planner at the arrival time of the order. We perform a simulation study to compare the performance of different policies under this assumption. Note that the exact FPT of each order is included in the order history of our data set, so that we can feed it into our simulation.

Impact of driving-time-constraint $\breve{d}(\cdot)$. We analyze how the driving-time constraint $\breve{d}$ impacts the KT policy. We simulate our policy with values $\breve{d} \in \{100, 200, 300, 400, 600, \infty\}$ seconds (which leads to driving distances of $\{0.5, 1, 1.5, 2, 3, \infty\}$ km with the average driver speed), and numerically optimize thickness level k^* for each value of $\breve{d}$. Note that the case of $\breve{d} = \infty$ refers to the case without a limit on driving time. In Table 1, we find that the total costs $W^{\Omega^{k^*,\breve{d}}}$ are at first strongly decreasing in $\breve{d}$, then slightly increasing, with $\breve{d}^* = 300$ being the optimal value ($W^{\Omega^{k^*,\breve{d}^*}} = 104$). The larger the neighborhood, the more opportunities for the order to be matched with a 'good' driver, thereby incurring less overtime penalty and reducing total costs. The protection against the assignment of far-away drivers is partially achieved by the market thickening mechanism. However, an oversized neighborhood can also lead to a slight increase in the costs due to matching with faraway drivers. This is consistent with the insights of Li and Netessine (2020): Range $\breve{d}$ helps the planner reduces unnecessary search friction and avoid certain costly drivers.

Table 1 **Results of the KT policy $\Omega^{k^*,\breve{d}}$ as a function of driving-time range $\breve{d}$ (for the overall scenario).**

$\breve{d}$ (sec)	100	200	300	400	600	∞
Opt. level k^*	4	4	5	5	5	5
Total Costs $W^{\Omega^{k^*,\breve{d}}}$	17827	520	104	108	110	110
Service Level V_l^{Ω}(%)	32.96	91.57	98.64	99.32	99.72	99.72
Drive-to-Shop time V_p^{Ω} (sec)	68	98	100	105	108	108

Accuracy of the simplified model. Next, we analyze the accuracy of the simplified model (Subsection 4.3). Given $\breve{d}$ as the driving-time limit, we set $\bar{\xi}$ to the average FPT and $\bar{d}$ to the non-compressible driving time computed from the KT policy with $k = 1$. Value $\tilde{k}$ is determined numerically with simulation. To determine value θ, we utilized the equation $\theta = \ln \tilde{k} / (\bar{\xi} - \bar{d})$ to reflect the relationship $(\bar{\xi} - \bar{d}) \approx \sum_{j=1}^{\tilde{k}} \frac{1}{\mu_j}$, and then we further numerically fine-tuned θ for better accuracy. Figure 6 shows the total costs obtained from the simulation of the KT policy and from the simplified model for all three scenarios (no-rush, rush, overall) and as a function of thickening level k. The average approximation gap in Figure 6, i.e., the mean absolute percentage error $\epsilon = |\widetilde{W}^{\Omega^{k,\check{d}}} - W^{\Omega^{k,\check{d}}}|/W^{\Omega^{k,\check{d}}}$, over all three scenarios is 5.17%. We conclude that our simplified model approximates the actual performance of the KT policy fairly accurately. Given Figure 6, we also conclude that the approximation model exhibits the same structural behavior as the KT policy, such as the quasi-convexity in k, shown in Proposition 4, and that it has similar optimal values k^*, indicating that the simplified model can be used to optimize parameter k.

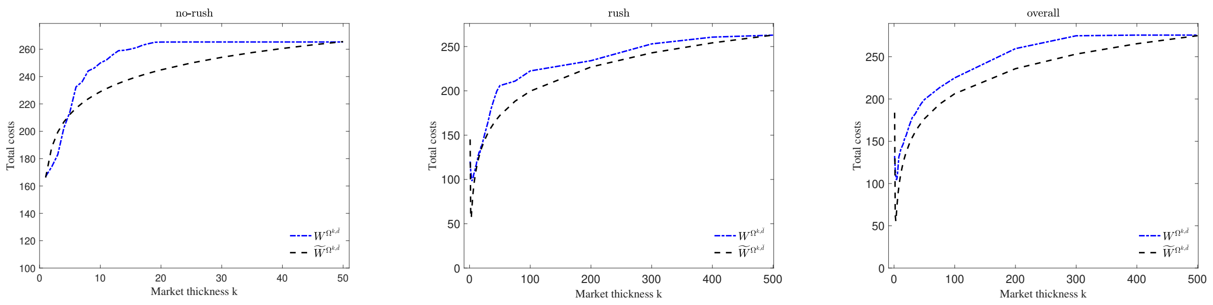


Figure 6 Total costs comparison of the KT policy and the simplified model for the three scenarios.

Table 2 Comparison of total costs of different matching policies.

	Greedy	Batching	MAR	KT	Lower bound
Scenarios	W^{Ω^g}	W^{Ω^b}	W^{Ω^r}	$W^{\Omega^{k^*,\check{d}^*}}$	W^{Ω^f}
no-rush	872	626	969	187	135
rush	1542	1198	813	98	66
overall	1346	1017	833	104	69

Performance comparison of different policies and lower bound. Table 2 shows the total costs obtained from the simulation of the different policies that we analyze. For the KT policy, we numerically optimized the thickness level k and the driving time constraint $\check{d}$, and obtained $k^* = 4$ and $\check{d}^* = 300$. In comparison to the benchmark policies, in the *overall* scenario, the KT policy has 92% lower total costs than the Greedy policy, 90% lower total costs than the Batching policy, and 88% lower costs than the MAR policy. The results indicate that the KT policy significantly outperforms the benchmark policies, which confirms the value of market thickening. In addition, we find that the KT policy has a performance close to the lower bound, W^{Ω^f} of the full-information case, indicating the effectiveness of the policy.

Table 3 shows performance indicators other than the total costs of the policies under investigation. For ease of exposition, we only provide the performance indicators for the *overall* scenario. The average drive-to-shop time V_p^{Ω} of the KT policy is comparatively shorter than those of the other policies such as Greedy and Batching, indicating the benefits of market thickening. In addition, we observe that the engagement time V_e^{Ω} per order of a driver is shorter under the KT policy than that under other policies, in particular compared to the Greedy and the Batching policies. As a result, the platform can considerably

Table 3 Performance indicators of different matching policies in the overall scenario.

Policies	Drive-to-Shop V_p^Ω	Delay V_d^Ω	Service Level V_l^Ω (%)	Engagement Time V_e^Ω	Engaged Drivers V_n^Ω	Waiting Time V_w^Ω	Idle Time V_q^Ω	Net-income V_s^Ω
Ω^g	127	2	99.18	2135	164	1209	1675	11.1
Ω^b	274	3	90.39	1708	131	634	3668	6.3
Ω^r	119	119	0	919	70	0	2068	6.0
$\Omega^{k^*,\check{d}^*}$	100	1	98.64	900	69	0	1596	7.1
Ω^f	69	0	100	869	67	0	1620	6.9

All time values in seconds.

increase the delivery capacity (V_n^Ω under the KT policy is approx. half than those of the Greedy and Batching policies). Furthermore, from V_w^Ω, we observe that the KT policy eliminates shop-waiting time, indicating the significance of finding an appropriate matching time to reduce waiting. The MAR policy also eliminates shop-waiting, but creates serious delay that has significant impact on the service level. The social welfare of drivers, represented by net income V_s^Ω and expected idle time V_q^Ω in Table 3 is highest under the Greedy policy, as the policy compensates drivers even if they are waiting. However, it achieves this outcome with high costs for the platform for waiting at restaurants. The KT policy, on the contrary, achieves comparably high net-income for drivers at competitive platform costs, leading to higher overall efficiency of the ecosystem than the other policies. The discussion shows that a matching policy has to trade-off the triangular objectives of three parties: platform (low cost), drivers (high income), and customers (high service level). The company has to define its balance to these parties. In Appendix 7.4.5, we propose a socially friendly KT policy that pays and releases drivers after a cut-off limit on idle time.

5.3. Performance Analysis with Unknown FPT

In Subsection 5.2, we considered the case where the platform knows the exact FPT at the time of assigning a driver to an order. This is, however, rarely the case in practice. In this subsection, we analyze how FPT uncertainty affects the different policies. First, we present a machine learning model to predict FPT values in real time. Then, we analyze the policy performance. We focus on the *overall* case in this section.

5.3.1. Prediction model for food processing times In recent years, the prediction of delivery time for online food delivery has received much attention (e.g., Gao et al. 2021, Liu et al. 2021b) but we are unaware of any work to predict FPT. For our problem, however, an accurate prediction of the FPT facilitates the matching between order and drivers, reduces 'shop-waiting' of drivers and delivery delays.

We propose a *causal* forecasting model, i.e., a model that exploits the characteristics of an order to predict its FPT. We compare several machine learning models to calibrate the prediction. To feed the FPT prediction procedure, we take the ∼120,000 orders of a complete month in the target region in the *overall* scenario. We use 80% of the order in-sample data to train the model, and the remaining 20% out-sample data to test the performance of the prediction model. First of all, we have to identify the characteristics that influence the FPT most. A technical report (Hao 2017) from scientists at Meituan indicate the following factors as key determinants for the FPT: Order arrival time (weekday, hour, minute of a day), food value, packaged box value and number of dishes (food number), order congestion situation of the food provider (number of waiting orders), food provider ID, and weather condition (rainy vs. sunny). To align the scale of features, we normalize the continuous features and convert the categorical features by the *one-hot encoding* technique (Brownlee 2018). Then, we test a wide range of machine learning methods including *eXtreme Gradient Boosting (XGBoost), k-Nearest Neighborhood (kNN), Lasso, Deep Neural Network (DNN), Random Forest (RndForest)* and *Ridge Regression (Ridge)*, see e.g., Hastie et al. (2009). We feed each method with the training data and conduct 10-fold cross-validation to select the best hyper-parameters for the models, for example, the learning rate, the shrinkage parameters and the number of trees. We implemented the training and testing procedures with *scikit-learn* package in Python.

Table 4 Average predicted FPT and FPT prediction errors of different machine learning methods (in minutes)

Method		XGBoost	kNN	Lasso	DNN	RndForest	Ridge
In-Sample	$\bar{\xi}$	19.57	19.51	19.55	19.50	19.55	19.55
	`RMSE`	7.25	8.19	8.88	9.12	7.73	7.89
	`MAE`	5.52	6.17	6.85	6.98	5.95	5.99
Out-of-Sample	$\bar{\xi}$	21.52	21.67	20.17	23.55	20.51	21.47
	`RMSE`	8.62	8.98	11.08	11.76	9.65	10.08
	`MAE`	6.58	6.84	8.41	9.35	7.35	7.76

The out-of-sample results of different measurements and methods are summarized in Table 4. We report the value of the average FPT $\bar{\xi}$, the square root of the mean squared error (`RMSE`) and the mean absolute error (`MAE`) for each method. The results indicate that XGBoost produces the smallest out-of-sample `RMSE`, 8.62, and hence, we adopt XGBoost for our numerical experiments. The four main factors that affect the predicted FPT by XGBoost are order arrival time (F-score 6,281; where the F-score is the number of times that a feature is used to split the data across the prediction process, indicating its relative importance), food value (F-score 5,322), the congestion situation/waiting number (F-score 1,854), and the packaged box value (F-score 959). Our predictors explain 49% of the total FPT variance ($R^2 = 0.49$). Note that several of the above factors (e.g., arrival time, food value, and order waiting number) may be proxies for the difficult-to-observe congestion level at the restaurant.

5.3.2. Performance analysis with predicted FPT. First, we analyze the additional costs from uncertainty in the FPT. Figure 7 shows the differences in total costs between the KT policy applied to the case of perfect FPT (see Subsection 5.2) and the BKT policy applied to the case of unknown FPT, under different protection levels $\hat{P}$. Thickness level k^* and $\breve{d}^*$ are numerically optimized for each

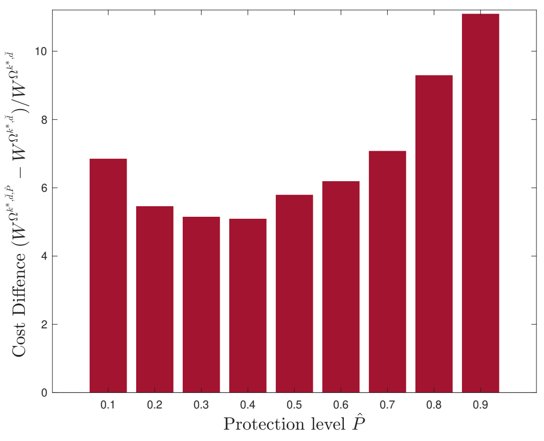


Figure 7 **Percentage difference in total costs between the KT policy with known FPT** $(\Omega^{k^*,\check{d}^*})$ **and the BKT with unknown FPT** $(\Omega^{k^*,\check{d}^*,\hat{P}})$ **for different protection levels** $\hat{P}$**.**

protection level. From Figure 7, we observe that the total costs in the case of unknown FPT decrease and then increase with the protection level (with $\hat{P}^* = 0.4$), indicating that it can even be advantageous to include a negative buffer (i.e., $\hat{P} < 0.5$). The results also indicate that ignoring the uncertainty leads to a considerable cost increase, sixfold for $\hat{P} = 0.5$.

Table 5 **Performance comparison for different policies and real-case (with unknown FPT for overall scenario).**

	Total Costs	Drive-to-Shop	Delay	Service Level	Engagement Time	Engaged Drivers	Waiting Time	Idle Time	Net-income
Policies	W^Ω	V_p^Ω	V_d^Ω	V_l^Ω (%)	V_e^Ω	V_n^Ω	V_w^Ω	V_q^Ω	V_s^Ω
Ω^g	1346	127	2	99.18	2135	164	1209	1675	11.1
Ω^b	1180	287	41	81.19	1731	133	645	3730	6.3
Ω^r	833	119	119	0	919	70	0	2068	6.0
$\Omega^{k^*,\check{d}^*,\hat{P}^*}$	634	110	39	64.42	1201	92	291	1522	8.7
real-case	1383	472	0	100	2183	167	693	–	19.8

All time values in seconds.

Table 5 shows total costs and other indicators for the BKT, Greedy, MAR, and Batching policies as well as for the real decisions that we observed in the data set from Meituan. The Meituan data set does not show delayed fetching, and because we know that the company uses a policy similar to the Greedy policy so we assume that the delay is close to zero. The BKT policy with numerically optimized parameters $k^* = 5, \check{d}^* = 300, \hat{P}^* = 0.4$ reduces total costs by 54% compared to the real decisions (and to the Greedy policy), by 24% compared to the MAR policy, and by 46% compared to the Batching policy. The BKT policy also dominates other policies for the drive-to-shop time V_Ω^p. The impact of the policies on driver net income is similar as the one we have observed for known FPT.

In Table 5, we have numerically searched for the optimal parameters $k^*, \check{d}^*, \hat{P}^*$ for the BKT policy. To provide some intuition on how total costs are affected by the parameters, Figure 8 plots total costs by thickness level k and protection level $\hat{P}$ for given $\check{d}^* = 300$ seconds. We do not only observe that total costs are quasi-convex in the thickening level k, confirming the result of Proposition 4, we also observe that total costs are quasi-convex in protection level $\hat{P}$ for a given thickening level k.

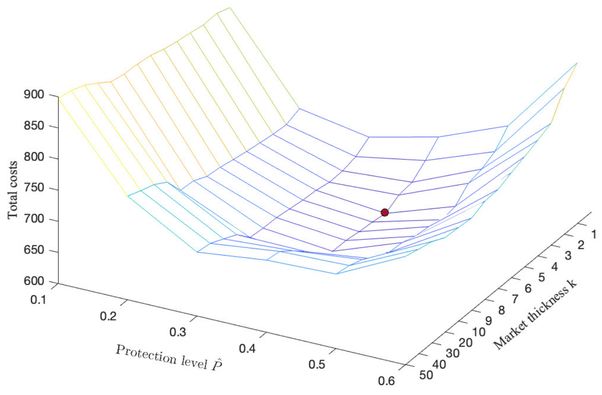


Figure 8 Total costs under BKT policy as a function of thickening and protection levels for given $\breve{d}^* = 300$.

Impact of prediction error σ_ξ. Next, we analyze the sensitivity of the BKT policy to the prediction error σ_ξ. From Table 2, we know that the KT policy is sensitive to FPT prediction errors. Therefore, we investigate the impact of σ_ξ on the performance of the BKT policy by performing experiments with different values of prediction error. In the simulation, we replace the FPT prediction with a weighted combination between the FPT prediction with XGBoost and the real FPT value from the data set (based on the *overall* scenario). Changing the weight of the exact FPT value in the weighted combination allows us to construct FPT predictions with varying degrees of σ_ξ. The corresponding results are shown in Figure 9. With given $\breve{d}^* = 300$, the results indicate that the BKT policy with numerically optimized parameters of k^* and $\hat{P}^*$ dominates the Batching policy for all different values of prediction error. Furthermore, the performance advantage of the BKT policy over the Batching policy increases significantly when the prediction error decreases (i.e., when the weight of the exact FPT in the prediction increases). This observation demonstrates the importance of accurate FPT prediction. In practice, OFD platforms can invest in better FPT prediction, for example, by collecting more information about food orders, such as food details and the number of employees currently on shift, or by providing incentives to restaurants to reduce the variability of FPT. The food platform may provide revenue sharing or rewards to restaurants for limiting FPT to a predefined range. Similar measures are used by online ride-hailing platforms to reduce variability of driving times. For example, drivers at platform *Uber* have to follow GPS-calculated routes to avoid low ratings and even a penalty (Liu et al. 2021a).

In Appendix 7.4, we analyze the relaxation of certain model assumptions, including scenarios with endogenous FPT, driver-type-specific compensation schemes, variable driver speed, regional clustering, driver abandonment, and a socially friendly adaptation of the BKT policy.

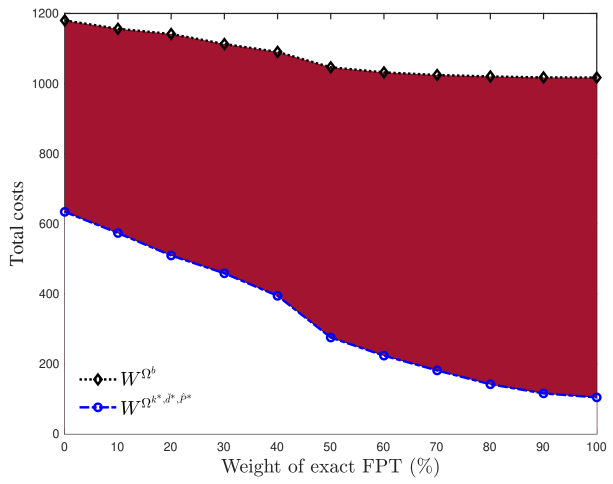


Figure 9 Total costs of the BKT policy and the Batching policy as a function of the precision of FPT prediction for given $\breve{d}^* = 300$.

6. Conclusion

In this paper, we have explored market thickening and dynamic order-driver matching on OFD platforms, drawing motivation from the operations of the Meituan platform in China. We have expanded upon existing matching solutions by incorporating uncertain food processing times into the matching decisions. We have formally defined a dynamic matching model as an SMDP and derived bounds on the performance based on different information schemes (known FPT vs. unknown FPT). In the case of a single order in isolation, we demonstrated that the optimal policy follows a threshold structure. Based on this result, for the general matching system, we have formulated a driving-time-constrained k-level thickening (KT) policy for real-time order and driver matching, and extended this policy to the Buffered k-level thickening (BKT) policy, which includes a risk buffer to safeguard against the prediction error of the FPT. We have developed a closed-form approximation of the KT policy's performance and derived several analytical properties of the approximation, including the quasi-convexity of the total costs with respect to the thickening level. Through extensive numerical experiments using a real data set from Meituan, we observed that the BKT policy reduces the drive-to-shop time by 77% and shop-waiting time by 58% compared to the real-case benchmark, effectively doubling driver efficiency. Similarly, under the BKT policy, the number of orders delivered increased by 1.35 units/hour, resulting in a reduction in the number of required drivers from 167 to 92. In conclusion, the BKT policy achieved a total costs reduction of 54% compared to the benchmark.

Our research yields several noteworthy insights. First, we ascertain that delaying the matching of orders during food processing to cultivate a thicker marketplace is advantageous. Strategies such as greedy matching, matching after the food is ready, and short-interval batching are significantly outperformed by the KT policy, which dynamically adapts the market thickness. Importantly, we note that the FPT is pivotal information for making matching decisions. Despite utilizing an advanced prediction model, a degree of prediction error persists. This error contributes to a substantial increase in costs when the policy is not adapted to accommodate the uncertainty of the FPT, even though, the cost increase can be significantly mitigated through the BKT policy. This observation suggests that OFD platforms benefit from investing in enhancing FPT prediction accuracy, possibly by gathering more information from restaurants or by offering incentives to restaurants to reduce FPT variability.

The solutions and insights derived from our analysis are not confined to OFD platforms but extend to any setting where dispatching and driving run concurrently with processing. This is observable, for instance, in one-to-one delivery platforms with time-sensitive targets, such as *Shansong* (ishansong.com) and *Lalamove* (huolala.cn). These platforms necessitate the processing of goods (e.g., packaging or loading) before they are ready for shipping. Container-trucking platforms like *Duckbill* (en.duckbillscm.com) present another analogous scenario. These platforms coordinate the delivery of containers to ports for international transportation, where the uncertain container loading times at the factories resemble the FPT in our study. The platform must judiciously assign containers to available trucks to ensure that neither the truck waits unnecessarily at the factory nor the container misses its scheduled customs clearance time at the port. Furthermore, our research opens several avenues for future exploration. In this paper, we have solely focused on one-to-one matching, notwithstanding the prevalent practice of consolidating multiple orders in-vehicle for drivers. This practice allows drivers to collect orders for multiple customers in a single pickup activity, implying that a matched driver might remain on the platform post-assignment. Another promising avenue is the real-time forecasting of driver speed, potentially employing machine learning techniques as suggested by Tao et al. (2022), and incorporating this information dynamically into the matching policy. Lastly, exploring the coordination between the restaurant's operational decisions and the platform's matching/routing decisions offers a compelling direction to further enhance efficiency.

Acknowledgments

References

Akbarpour, Mohammad, Shengwu Li, Shayan Oveis Gharan. 2020. Thickness and information in dynamic matching markets. *Journal of Political Economy* **128**(3) 783–815.

Aouad, A, O Saritac. 2022. Dynamic stochastic matching under limited time. *Operations Research* **70**(4) 2349–2383.

Arnosti, Nick, Ramesh Johari, Yash Kanoria. 2021. Managing congestion in matching markets. *Manufacturing & Service Operations Management* **23**(3) 620–636.

Ashlagi, Itai, Maximilien Burq, Patrick Jaillet, Amin Saberi. 2018. Maximizing efficiency in dynamic matching markets. *arXiv preprint arXiv:1803.01285* .

Banerjee, Siddhartha, Daniel Freund, Thodoris Lykouris. 2016. Pricing and optimization in shared vehicle systems: An approximation framework. *arXiv preprint arXiv:1608.06819* .

Banerjee, Siddhartha, Yash Kanoria, Pengyu Qian. 2018. State dependent control of closed queueing networks with application to ride-hailing. *arXiv:1803.04959v2* .

Baxendale, Peter. 2011. T. e. harris's contributions to recurrent markov processes and stochastic flows. *The Annals of Probability* **39**(2). doi:10.1214/10-aop594. URL `https://doi.org/10.1214%2F10-aop594`.

Bosredon, Mickael. 2021. Deliveroo : Flexibilité, protection sociale... la question du statut des livreurs á vélo en débat. `https://www.20minutes.fr/economie/`.

Brownlee, Jason. 2018. Why one-hot encode data in machine learning? `https://machinelearning mastery.com/why-one-hot-encode-data-in-machine-learning/`.

Cachon, G.P., K.M. Daniels, R. Lobel. 2017. The role of surge pricing on a service platform with self-scheduling capacity. *Manufacturing & Service Operations Management* **19**(3) 368–384.

Cao, Yufeng, Anton Kleywegt, He Wang. 2020. Dynamic pricing for truckload transportation marketplaces. *Available at SSRN 3700227* .

Carlsson, John Gunnar, Sheng Liu, Nooshin Salari, Han Yu. 2021. Provably good region partitioning for on-time last-mile delivery. *Available at SSRN 3915544* .

Castro, Francisco, Peter Frazier, Hongyao Ma, Hamid Nazerzadeh, Chiwei Yan. 2021. Matching queues, flexibility and incentives.

Chen, Manlu, Ming Hu, Jianfu Wang. 2022. Food delivery service and restaurant: Friend or foe? *Management Science* **68**(9) 6539–6551.

Chen, Mingliu. 2019. Matching supply and demand with mismatch-sensitive players. *Available at SSRN 3458673* .

Chen, Mingliu, Ming Hu. 2020. Courier dispatch in on-demand delivery. *Available at SSRN 3675063* .

Chen, Ying-Ju, Tinglong Dai, C Gizem Korpeoglu, Ersin Körpeoğlu, Ozge Sahin, Christopher S Tang, Shihong Xiao. 2020. Om forum—innovative online platforms: Research opportunities. *Manufacturing & Service Operations Management* **22**(3) 430–445.

Cohen, Maxime C, Michael D Fiszer, Baek Jung Kim. 2022. Frustration-based promotions: Field experiments in ride-sharing. *Management Science* **68**(4) 2432–2464.

Cui, Ruomeng, Wenchang Zhang, Zhanzhi Zheng. 2022. Market thickness and delivery efficiency in food delivery platforms. *Available at SSRN 4239864* .

Curry, David. 2021. Food delivery app revenue and usage statistics (2021). `https://www.businessofapps.com/data/food-delivery-app-market/`.

Gan, Li, Qi Li. 2016. Efficiency of thin and thick markets. *Journal of Econometrics* **192**(1) 40–54.

Gao, Chengliang, Fan Zhang, Guanqun Wu, Qiwan Hu, Qiang Ru, Jinghua Hao, Renqing He, Zhizhao Sun. 2021. A deep learning method for route and time prediction in food delivery service. *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 2879–2889.

Gemmell, Katharine. 2021. Deliveroo's stock market debut is a flop. should you still buy? `https://www.bloomberg.com/news/articles/2021-03-31/`.

Gupta, Ashutosh. 2022. What is a supply chain control tower - and what's needed to deploy one? `https://www.gartner.com/en/articles/what-is-a-supply-chain-control-tower-and-what-s-`
needed-to-deploy-one.

Hao, Jinghua. 2017. Real-time delivery order assignment policy: modeling and optimization. `https://tech.meituan.com/2017/10/11/o2o-intelligent-distribution.html`.

Hasija, Sameer, Zuo-Jun Max Shen, Chung-Piaw Teo. 2020. Smart city operations: Modeling challenges and opportunities. *Manufacturing & Service Operations Management* **22**(1) 203–213.

Hastie, Trevor, Robert Tibshirani, Jerome Friedman. 2009. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media.

Hirschberg, C, A Rajko, T Schumacher, Wrulich M. 2016. The changing market for food delivery. https://www.mckinsey.com/industries/technology-media-and-telecommunications/our-insights/the-changing-market-for-food-delivery.

Hu, Ming, Yun Zhou. 2022. Dynamic type matching. *Manufacturing & Service Operations Management* **24**(1) 125–142.

Levi Sumagaysay. 2020. The pandemic has more than doubled food-delivery apps' business. Now what? https://www.marketwatch.com/story/.

Li, Jun, Serguei Netessine. 2020. Higher market thickness reduces matching rate in online platforms: Evidence from a quasiexperiment. *Management Science* **66**(1) 271–289.

Liu, Meng, Erik Brynjolfsson, Jason Dowlatabadi. 2021a. Do digital platforms reduce moral hazard? the case of uber and taxis. *Management Science* **67**(8) 4665–4685.

Liu, Sheng, Long He, Zuo-Jun Max Shen. 2021b. On-time last-mile delivery: Order assignment with travel-time predictors. *Management Science* **67**(7) 4095–4119.

Liu, Tracy Xiao, Zhixi Wan, Chenyu Yang. 2019. The efficiency of a dynamic decentralized two-sided matching market. *Available at SSRN 3339394* .

Manshadi, Vahideh, Scott Rodilitz, Daniela Saban, Akshaya Suresh. 2022. Online algorithms for matching platforms with multi-channel traffic. *SSRN Electronic Journal* doi:10.2139/ssrn.4036904. URL https://doi.org/10.2139%2Fssrn.4036904.

Mao, Wenzheng, Liu Ming, Ying Rong, Christopher S Tang, Huan Zheng. 2019. Faster deliveries and smarter order assignments for an on-demand meal delivery platform. *Available at SSRN 3469015* .

Mao, Wenzheng, Liu Ming, Ying Rong, Christopher S Tang, Huan Zheng. 2022. On-demand meal delivery platforms: Operational level data and research opportunities. *Manufacturing & Service Operations Management* **0**(0).

McLaren, John. 2003. Trade and market thickness: effects on organizations. *Journal of the European Economic Association* **1**(2-3) 328–336.

Meituan. 2021. Let more voices participate in the change, the meituan takeaway "order allocation" algorithm is public (in chinese). https://mp.weixin.qq.com/s/qyegF_r_SPGnkEdZqkVjxA.

Meituan, Dianping. 2020. Annual report 2019. http://meituan.todayir.com/html/index.php.

Özkan, Erhun, Amy R Ward. 2020. Dynamic matching for real-time ride sharing. *Stochastic Systems* **10**(1) 29–70.

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

Roth, Alvin E. 2015. *Who gets what–and why: the new economics of matchmaking and market design*. Houghton Mifflin Harcourt.

Ryan, Waliany, Kang Lei, Murati Ernel, Amin Mohammad Shafkat. 2018. How trip inferences and machine learning optimize delivery times on uber eats. https://www.uber.com/en-GB/blog/uber-eats-trip-optimization/.

Shaked, Moshe, George Shanthikumar. 1988. Stochastic convexity and its applications. *Advances in Applied Probability* **20** 427–446.

Sun, Ping. 2019. Your order, their labor: An exploration of algorithms and laboring on food delivery platforms in china. *Chinese Journal of Communication* **12**(3) 308–323.

Tao, Jiawei, Hongyan Dai, Weiwei Chen, Hai Jiang. 2022. The value of personalized dispatch in o2o on-demand delivery services. *European Journal of Operational Research* **0**(0).

Topkis, Donald M. 1978. Minimizing a submodular function on a lattice. *Operations Research* **26**(2) 305–321.

Ulmer, Marlin W, Barrett W Thomas, Ann Melissa Campbell, Nicholas Woyak. 2021. The restaurant meal delivery problem: dynamic pickup and delivery with deadlines and random ready times. *Transportation Science* **55**(1) 75–100.

Varma, Sushil Mahavir, Pornpawee Bumpensanti, Siva Theja Maguluri, He Wang. 2022. Dynamic pricing and matching for two-sided queues. *Operations Research* **0**(0).

Voccia, Stacy A, Ann Melissa Campbell, Barrett W Thomas. 2019. The same-day delivery problem for online purchases. *Transportation Science* **53**(1) 167–184.

Xie, Yaqi, Will Ma, Linwei Xin. 2022. The benefits of delay to online decision-making. *Available at SSRN* .

Yan, Chiwei, Helin Zhu, Nikita Korolko, Dawn Woodard. 2020. Dynamic pricing and matching in ride-hailing platforms. *Naval Research Logistics (NRL)* **67**(8) 705–724.

Zehtabian, Shohre, Christian Larsen, Sanne Wøhlk. 2022. Estimation of the arrival time of deliveries by occasional drivers in a crowd-shipping setting. *European Journal of Operational Research* **0**(0).

7. Appendix

7.1. Additional performance indicators

We analyze performance of our policies beyond total costs. In this appendix, we introduce additional performance indicators. Traditionally, allocation policies focus on the expected *drive-to-shop time* per order, which we define for a given policy Ω as

$$V_p^\Omega = \lim_{t\to\infty} \mathbb{E} \tfrac{1}{|\mathcal{I}_t|} \textstyle\sum_{(i,j,t_{ij})\in\Omega_t} d_{ij}(t_{ij}). \tag{15}$$

Obviously, small V_p^Ω indicates an assignment of drivers near the restaurants. In our paper, we also consider the expected *waiting-in-shop time per order* that drivers wait at the restaurant,

$$V_w^\Omega = \lim_{t\to\infty} \mathbb{E} \tfrac{1}{|\mathcal{I}_t|} \textstyle\sum_{(i,j,t_{ij})\in\Omega_t} (\tau_i + \xi_i - t_{ij} - d_{ij}(t_{ij}))^+, \tag{16}$$

which allows us to define the expected *total engagement time per order*, from matching to final delivery, V_e^Ω, as $V_e^\Omega = V_p^\Omega + V_w^\Omega + \overline{\psi}$.

To analyze the service quality, we consider indicators such as expected *delay per order*,

$$V_d^\Omega = \lim_{t\to\infty} \mathbb{E} \tfrac{1}{|\mathcal{I}_t|} \textstyle\sum_{(i,j,t_{ij})\in\Omega_t} \left(t_{ij} + d_{ij}(t_{ij}) - \tau_i - \xi_i\right)^+, \tag{17}$$

and *service level*,

$$V_l^\Omega = 1 - \lim_{t\to\infty} \mathbb{E} \tfrac{1}{|\mathcal{I}_t|} \textstyle\sum_{(i,j,t_{ij})\in\Omega_t} \mathbf{1}\left(t_{ij} + d_{ij}(t_{ij}) - \tau_i - \xi_i > 0\right), \tag{18}$$

where $\mathbf{1}(\cdot)$ is the indicator function. Low V_d^Ω and high V_l^Ω indicate a high service quality.

We use two different indicators to measure the *efficiency of driver allocation or usage*:

$$\text{Expected idle time per driver}: V_q^\Omega = \lim_{t\to\infty} \mathbb{E} \frac{1}{|\mathcal{I}_t|} \sum_{(i,j,t_{ij})\in\Omega_t} (t_{ij} - \tau_j), \tag{19}$$

$$\text{Expected number of engaged drivers}: V_n^\Omega = \lambda_o \cdot V_e^\Omega, \tag{20}$$

where V_q^Ω describes the expected waiting-to-be-matched time of a driver. Short V_q^Ω implies short waiting for a driver to be assigned to an order, which is an important concern for participation of self-scheduled drivers in platform scheduling (Cachon et al. 2017). V_n^Ω indicates the total number of drivers engaged on average and relies on Little's Law.

The matching decisions have a significant impact on the number of orders that a driver can complete, ultimately affecting their earning potential (Ryan et al. 2018). Finally, as a reflection of drivers' social welfare, we analyze the expected *net-income per time unit of a driver*, which we define as:

$$V_s^\Omega = \tfrac{c_e \cdot V_e^\Omega - c_g \cdot v \cdot (V_p^\Omega + \overline{\psi})}{V_e^\Omega + V_q^\Omega}. \tag{21}$$

V_s^Ω can be interpreted as the ratio of a driver's total income $c_e \cdot V_e^\Omega$ (where c_e denotes the reward per unit time) minus driving cost $c_g \cdot v \cdot (V_p^\Omega + \overline{\psi})$ (where c_g is the cost per mileage, e.g., for gasoline, and v is the average driving speed) to the *work duration* of a driver, $V_e^\Omega + V_q^\Omega$. Indicator V_s^Ω reflects the net-income per time unit invested by a driver and is an important indication for the economic attractiveness of a matching policy for drivers.

7.2. SMDP model formulation

We formally define our model with known FPT (see Section 3) as an SMDP by describing its elements.

State: Even though the arrival processes are continuous, we consider that decisions are only made upon driver or order arrivals, driver abandonment, or edge expiration times. Hence, we can write $\mathbf{S}_n = (\mathbf{x}_n, \mathbf{y}_n, e_n, \tilde{t}_n)$ for the state that the planner observes at the time of the n^{th} event time. The set $\mathbf{x}_n = \{((\xi_i')_n, l_i)\}$ records information of all arrived but unmatched orders. Let $(\xi_i')_n = \xi_i + \tau_i - \tilde{t}_n$ represent the 'remaining' FPT of an open order i at the current time $\tilde{t}_n$, and l_i the order's restaurant location. Note that we formulate the SMDP based on 'remaining' times, using the prime ($'$) to distinguish remaining durations from absolute time epochs. $(\xi_i')_n$ can take negative value indicating that it already exceeds the FPT but the order is still not matched. $\tilde{t}_n$ is the 'time index', i.e., the current time in the periodicity δ of the time-dependent driving times. The set $\mathbf{y}_n = \{l_j\}$ indicates the location of each active driver j (note that we do not need to keep track of drivers' abandonment times as these are not known to the planner before the abandonment). Finally, the event indicator $e_n \in \{0,1,2,3\}$ indicates the current decision epoch occurs when an order arrives ($e_n = 0$), a driver arrives ($e_n = 1$), a compatible edge expires ($e_n = 2$), or a driver abandons ($e_n = 3$). We denote the state space of all possible $\mathbf{S}_n$ by set $\mathcal{S}$.

Actions and costs: If the planner decides to match an order i with a driver j, $\mathbf{a}_n = (i,j)$; otherwise, no action is taken and $\mathbf{a}_n = \emptyset$. The action space is presented as $\mathbf{A}(\mathbf{S}_n)$ and contains all *feasible* matchings of orders and drivers. For a state $\mathbf{S}$ and action $\mathbf{a}$, the immediate cost is given by $C(\mathbf{S}_n, \mathbf{a}_n) = d_{ij}(\tilde{t}_n) + ((\xi_i')_n - d_{ij}(\tilde{t}_n))^+ + c \cdot (d_{ij}(\tilde{t}_n) - (\xi_i')_n)^+$ if $\mathbf{a}_n = (i,j)$ and $C(\mathbf{S}_n, \mathbf{a}_n) = 0$ if $\mathbf{a}_n = \emptyset$.

State transitions: For any state transition, the remaining FPT values are updated with sojourn time t_n^s by setting $(\xi_i')_n = (\xi_i')_{n-1} - t_n^s$ and time index $\tilde{t}_n$ is updated as $\tilde{t}_n = (\tilde{t}_{n-1} + t_n^s) \mod \delta$. If a driver abandons ($e = 3$), the driver is removed from the set of active drivers, i.e., $\mathbf{y}_n = \mathbf{y}_{n-1} \setminus \{l_j\}$. If the action $\mathbf{a}_n = (i,j)$ is selected, the planner will remove the corresponding elements respectively, i.e. $\mathbf{x}_n^+ = \mathbf{x}_n \setminus \{((\xi_i')_n, l_i)\}$ and $\mathbf{y}_n^+ = \mathbf{y}_n \setminus \{l_j\}$. At each transition, we can calculate the 'remaining' time to the expiration of an order-driver pair (i,j), i.e., $({t_{ij}^c}')_n$ from state $\mathbf{S}_n$, by $({t_{ij}^c}')_n = (\xi_i')_n - d_{ij}(({t_{ij}^c}')_n + \tilde{t}_n)$.

State transitions depend on the sojourn time and probabilities to transit from a stage $\mathbf{S}_n$ to a stage $\mathbf{S}_{n+1}$ if action $\mathbf{a}_n$ is taken. The transition process is not purely Markovian because the remaining edge expiration times ${t_{ij}^c}'$ are known and deterministic, only the order and driver arrival times and driver abandonment times are Markovian. Therefore, to describe the state transition behavior, we first need to characterize the random variable sojourn time $T^s(\mathbf{S}_n, \mathbf{a}_n)$ (with realization t_n^s), which is distributed according to cumulative distribution function

$$F(t^s|\mathbf{S},\mathbf{a}) = \begin{cases} \dfrac{1-e^{-(\lambda_o+\lambda_d+\lambda_a|\mathbf{y}|)t^s}}{1-e^{-(\lambda_o+\lambda_d+\lambda_a|\mathbf{y}|)\min\limits_{(i,j)}\left({t_{ij}^c}'|{t_{ij}^c}'>0\right)}}, & t^s < \min\limits_{(i,j)}\left({t_{ij}^c}'|{t_{ij}^c}' > 0\right), \\ 1, & t^s \geq \min\limits_{(i,j)}\left({t_{ij}^c}'|{t_{ij}^c}' > 0\right), \end{cases} \tag{22}$$

where $|\mathbf{y}|$ is the number of unmatched drivers and $\min_{(i,j)} \left(t^c_{ij}{}' | t^c_{ij}{}' > 0\right)$ is the minimum of the positive, feasible 'remaining' times to expiration of pairs $(i,j) \in \mathbf{A}(\mathbf{S}_n)$. If there is no pair (i,j) with $t^c_{ij}{}' > 0$, $F(t^s|\mathbf{S},\mathbf{a})$ reduces to $F(t^s|\mathbf{S},\mathbf{a}) = 1 - e^{-(\lambda_o+\lambda_d+\lambda_a|\mathbf{y}|)t^s}$.

Based on the distributions of drivers and orders, and given that arrival times and locations are independent, we can formulate the transition probabilities $p_{\mathbf{S}_n,\mathbf{S}_{n+1}}(\mathbf{a}_n|t^s_{n+1})$ from the above elements, which we omit for space considerations (details are available from the authors on request). The minimal expected cost per order, $W^*(\mathbf{S}_0)$, with initial state $\mathbf{S}_0$ given by

$$W^*(\mathbf{S}_0) = \lim_{\tilde{n}\to\infty} \min_{\mathbf{a}} \mathbb{E}\left[\frac{\sum_{n=1}^{\tilde{n}} C(\mathbf{S}_n, \mathbf{a}_n)}{\sum_{n=1}^{\tilde{n}} \mathbf{1}(e_n = 0)}\right], \tag{23}$$

where $\mathbf{1}(\cdot)$ is the indicator function. From the assumptions $\lambda_d > \lambda_o$ and $\lambda_a > 0$ and the definition of the SMDP follows that at least for a policy that immediately matches order-driver-pairs, the embedded Markov chain of the SMDP is ϕ-irreducible, positive Harris-recurrent, and, hence, has a stationary distribution (Baxendale 2011).

7.3. Proofs

Proof of Proposition 1. If an edge is matched and it is not expired, according to the objective function, the planner is always weakly better off if she waits and matches them when the edge expires. If there is no edge in the graph, it means there are only orders (or drivers) in the system. When the next feasible driver (or order) arrives, it will establish a matching with an order either at a compatible edge expiring or instantly, because the edge with the new arrival is already expired. Note that because of the assumption $t + d_{ij}(t) < t' + d_{ij}(t')$ for any $t < t'$, any edge has only a single edge expiring time. ■

Proof of Proposition 2. Using the cost function in Equation (2), we have cost of matching the single order with driver j with remaining food processing duration ξ^r (recall that we drop order index i from the notation when considering the single-order-model) of $d_j + (\xi^r - d_j)^+ + c\cdot(d_j - \xi^r)^+$. From the above equation, we can make two observations: matching at the edge expiration time $\xi^r = d_j$ minimizes the cost of matching driver j to the single order, and if we consider only edge expiration times for matching, the best match among a given driver pool is the one with the shortest driving time d_j.

From the further observation that the driver with the shortest driving time is also the one for which the edge expires last, we can follow that it is never optimal to match a driver if one or more edges are not yet expired, which finalizes the proof. ■

Proof of Theorem 1. Parts (a1) and (a2): This proof first establishes two key arguments of the threshold structure of the optimal decision: First, we show that in the single-order-model the optimal policy decides only at two different points in time: At driver arrivals it chooses the better between the best existing driver and the incoming driver to 'keep in the system'. At edge expiration, the system decides between matching the expiring driver and waiting for a better driver. The resulting stochastic process iterates over these times. Second, we show that for the decision at edge expiration, the future cost only depends on the remaining FPT and not on any other elements of the state space such as prior driver arrivals. This implies the threshold structure of the decisions. In the proof, we rely in parts on the formulation of the problem as an SMDP, as presented in Appendix 7.2.

From Proposition 2, we know that it is never optimal to allocate a driver to the single order at times other than the latest edge expiration time of existing drivers or the arrival of a new driver (if the edge is already expired at the time of arrival and there are no other non-expired edges). Hence, we can simplify the state $\mathbf{S}_n$ in our single-order system to $\mathbf{S}_n = (d_j, \xi^r)$ with driver j being the one whose edge expires last (the driver with the shortest travel time), and ξ^r being the remaining time until the food is prepared. Please note that before the arrival of the first driver, the state element d_j is naturally empty. In this case, we set $d_j = +\infty$.

Let us denote the expected minimum cost in state (d_j, ξ^r) at arrival time of driver j' (or at edge expiration time of driver j, $d_j = \xi^r$, in which case we set $d_{j'} = +\infty$) as $W(d_j, \xi^r | d_{j'})$. Furthermore, we use notation $W^+(d_j, \xi^r)$ to denote the expected minimum *future* cost right *after* a decision which results in the state (d_j, ξ^r). We can then construct function $W(d_j, \xi^r | d_{j'})$ from the following complete list of possible events.

(a) No arrival, but edge of existing driver j expires ($d_{j'} = \infty \wedge d_j = \xi^r$). In this case, it is optimal to *match or wait*, i.e., $W(d_j, \xi^r | d_{j'}) = \min\left\{d_j, W^+(d_j, \xi^r)\right\} = \min\left\{d_j + c\,(d_j - \xi^r)^+, W^+(d_j, \xi^r)\right\}$.

(b) Arrival of driver j' with expired edge and edge of existing driver j expired ($d_{j'} > \xi^r \wedge d_j > \xi^r$). In this case, it is also optimal to *match or wait*, i.e., $W(d_j, \xi^r | d_{j'}) = \min\left\{\min\{d_j + c\,(d_j - \xi^r)^+, d_{j'} + c\,(d_{j'} - \xi^r)^+\}, W^+(\min\{d_j, d_{j'}\}, \xi^r)\right\}$.

(c) Arrival of driver j' with expired edge and edge of existing driver j not expired ($d_{j'} > \xi^r \wedge d_j < \xi^r$). In this case, it is optimal to continue waiting with the non-expired edge, i.e., $W(d_j, \xi^r | d_{j'}) = W^+(d_j, \xi^r) = W^+(\min\{d_j, d_{j'}\}, \xi^r)$.

(d) Arrival of driver j' with non-expired edge and edge of existing driver j expired ($d_{j'} < \xi^r \wedge d_j > \xi^r$). In this case, it is optimal to wait with new driver and its non-expired edge, i.e., $W(d_j, \xi^r | d_{j'}) = W^+(d_{j'}, \xi^r) = W^+(\min\{d_j, d_{j'}\}, \xi^r)$.

(e) Arrival of driver j' with non-expired edge and edge of existing driver j not expired ($d_{j'} < \xi^r \wedge d_j < \xi^r$). In this case, it is optimal to wait, with the edge that expires last, i.e., $W(d_j, \xi^r | d_{j'}) = W^+(\min\{d_j, d_{j'}\}, \xi^r)$.

The expected *future* cost right after a decision and in state (d_j, ξ^r), $W^+(d_j, \xi^r)$, is given by taking the expectation over the next arrival:

$$W^+(d_j,\xi^r) = \mathbb{E}_{d_{j'}}\mathbb{E}_{\hat{t}_{j'}} \begin{cases} W(d_j,\xi^r - d_j|\infty), & \text{if } (\xi^r - d_j) \in \left[0,\hat{t}_{j'}\right], \\ W(d_j,\xi^r - \hat{t}_{j'}|d_{j'}), & \text{else,} \end{cases} \tag{24}$$

where j' is the next driver arrival with driving time $d_{j'}$ and exponential inter-arrival time $\hat{t}_{j'}$. Notably, the case $(\xi^r - d_j) \in \left[0,\hat{t}_{j'}\right]$ represents the event that the edge of driver j expires before the arrival of driver j'. Specifically, the equation can be enriched as

$$W^+(d_j,\xi^r) = \mathbb{E}_{d_{j'}}\mathbb{E}_{\hat{t}_{j'}} \begin{cases} \min\left\{d_j, W^+(\infty,\xi^r - d_j)\right\}, & \text{if } (\xi^r - d_j) \in \left[0,\hat{t}_{j'}\right], \\ W^+(d_j,\xi^r - \hat{t}_{j'}), & \text{if } (\xi^r - d_j) > \hat{t}_{j'} \text{ and } d_{j'} > (\xi^r - \hat{t}_{j'}), \\ W^+(\min\{d_j,d_{j'}\},\xi^r - \hat{t}_{j'}), & \text{if } (\xi^r - d_j) > \hat{t}_{j'} \text{ and } d_{j'} \le (\xi^r - \hat{t}_{j'}), \\ \min\left\{\min\{d_j + c\left[d_j - \xi^r\right]^+, d_{j'} + c\left[d_{j'} - \xi^r\right]^+\}, W^+(\infty,\xi^r - \hat{t}_{j'})\right\}, & \\ & \text{if } (\xi^r - d_j) < 0 \text{ and } d_{j'} > (\xi^r - \hat{t}_{j'}), \\ W^+(d_{j'},\xi^r - \hat{t}_{j'}), & \text{if } (\xi^r - d_j) < 0 \text{ and } d_{j'} \le (\xi^r - \hat{t}_{j'}), \end{cases} \tag{25}$$

Please note that, without loss of generality, we abused notation by writing the expectation $\mathbb{E}_{d_{j'}}$ for the expected travel time resulting from driver location $l_{j'}$.

The decision between matching the current driver and waiting for the future only occurs in case (a), and in case (b) if $d_{j'} < d_j$. For these two cases, the decision equation is the same, $\min\left\{d_j + c\left[d_j - \xi^r\right]^+, W^+(d_j,\xi^r)\right\}$, with j replaced by j' in the second case.

To show that the threshold with which drivers are matched depends only on the remaining FPT, we have to show that future cost are independent of prior driver arrivals. In order to show this result, we need the result that the optimal policy will never match the single order with an existing driver with an expired edge. Recall that an existing driver can only have an expired edge if it has not been matched at the time the edge expired (or the driver arrival time, whatever was later), i.e., if condition $d_j + c(d_j - \xi^r)^+ > W^+(d_j,\xi^r)$ was true at time $d_j = \xi^r$ (or the arrival time if the edge was already expired at arrival, $d_j > \xi^r$). From this time onwards, term $d_j + c(d_j - \xi^r)^+ = (c+1)d_j - c\xi^r$ is increasing at rate c in time. Term $W^+(d_j,\xi^r)$, on the other hand, can increase *at most* at rate c in time because of the following argumentation: For any sample path realization, the total costs $W^+(d_j,\xi^r)$ contain only a single cost term for a single match, given by structure $d_{j'} + c(d_{j'} - \xi^r)^+$ of some future driver j'. Thus, the maximum rate by which $W^+(d_j,\xi^r)$ can decrease in ξ^r, i.e., increase in time, is c, and condition $d_j + c(d_j - \xi^r)^+ > W^+(d_j,\xi^r)$ will always stay true. Hence, it is never optimal to match the order with this driver after the edge expiration time. From the fact that the optimal policy never matches an existing driver with an expired edge, we conclude that $W^+(d_j,\xi^r)$, at the time an edge expires or a driver with an expired edge arrives, is independent of d_j, i.e., $W^+(d^1,\xi^r) = W^+(d^2,\xi^r)$ for any two values $d^1 > \xi^r, d^2 > \xi^r$. Given this result, we can write without loss of generality $W^+(d_j,\xi^r) = W^+(\infty,\xi^r)$ for any $d_j > \xi^r$.

The decision given by expression $\min\left\{d_j + c\,[d_j - \xi^r]^+, W^+(\infty,\xi^r)\right\}$ has a threshold structure because the first term is obviously increasing in d_j while the second term is independent of d_j, such that both terms can become equal at *at most* a single point/interval. From the above argument also follows that case (b) in which $d_j < d_{j'}$ will never have a match as optimal decision.

Following this structure, and because we focus on expiring/expired edges for which $d_j \geq \xi^r$, the threshold can be derived from the following equation $d_j + c(d_j - \xi^r) \overset{!}{=} W^+(\infty,\xi^r)$. Equation (6) and the result of the theorem then follow directly as a consequence by setting

$$\tilde{d}(\xi^r) = \frac{W^+(\infty,\xi^r) + c\xi^r}{c+1}. \tag{26}$$

Part (b): From Equation (26) we observe that threshold function $\tilde{d}(\xi^r)$ is proportional to the sum of $W^+(\infty,\xi^r)$ and $c\xi^r$. In the proof of part (a) of this theorem, we have shown that $W^+(\infty,\xi^r)$ is independent of d_j and that it cannot decrease in ξ^r at a rate greater than c. Term $c\xi^r$ is obviously increasing in ξ^r at rate c. Hence, threshold function $\tilde{d}(\xi^r)$ as being proportional to the sum of $W^+(\infty,\xi^r)$ and $c\xi^r$ is non-decreasing in ξ^r. ■

Proof of Proposition 3. The pdf of a hypoexponential distribution is $\tilde{f}_k(x) = \sum_{j=k}^{\tilde{k}} \ell_j(0)\mu_j e^{-\mu_j x}$, where $\ell_j(0)$ is the Lagrange basis polynomial associated with point μ_j. It follows directly from Equation (2) that total costs are equivalent to

$$\begin{aligned}
\widetilde{W}^{\Omega^{k,\check{d}(\cdot)}}(k) &= \overline{d} + \int_0^{\overline{\xi}-\overline{d}} (\overline{\xi} - x - \overline{d}) \cdot \tilde{f}_k(x)dx + c \cdot \int_{\overline{\xi}-\overline{d}}^{\infty} (x + \overline{d} - \overline{\xi}) \cdot \tilde{f}_k(x)dx \\
&= \overline{d} + \int_0^{+\infty} (\overline{\xi} - x - \overline{d}) \cdot \tilde{f}_k(x)dx + (1+c) \int_{\overline{\xi}-\overline{d}}^{\infty} (x + \overline{d} - \overline{\xi}) \cdot \tilde{f}_k(x)dx \\
&= \overline{d} + \int_0^{+\infty} (\overline{\xi} - \overline{d}) \cdot \tilde{f}_k(x)dx - \int_0^{+\infty} x\tilde{f}_k(x)dx + (1+c) \int_{\overline{\xi}-\overline{d}}^{\infty} (x + \overline{d} - \overline{\xi}) \cdot \tilde{f}_k(x)dx \\
&= \overline{\xi} - \sum_{j=k}^{\tilde{k}} \mu_j^{-1} + (1+c)p(k)
\end{aligned} \tag{27}$$

with

$$p(k) = \int_{\overline{\xi}-\overline{d}}^{\infty} (x + \overline{d} - \overline{\xi}) \cdot \sum_{j=k}^{\tilde{k}} \mu_j e^{-x\mu_j} \left(\prod_{i=k,i\neq j}^{\tilde{k}} \frac{\mu_i}{\mu_i - \mu_j} \right) dx = \sum_{j=k}^{\tilde{k}} \frac{e^{-\mu_j(\overline{\xi}-\overline{d})}}{\mu_j} \left(\prod_{i=k,i\neq j}^{\tilde{k}} \frac{\mu_i}{\mu_i - \mu_j} \right). \tag{28}$$

Next, we transform expression $p(k)$ to allow us to evaluate it easily for varying values of k. We denote each element of the sum (on the right-hand side) in $p(k)$ as ϕ_j. Then, setting $j = k + m (0 \leq m \leq \tilde{k} - k)$, we obtain

$$\phi_j = \phi_{k+m} = \frac{e^{-\mu_{k+m}(\overline{\xi}-\overline{d})}}{\mu_{k+m}} \left(\prod_{i=k,i\neq k+m}^{\tilde{k}} \frac{\mu_i}{\mu_i - \mu_{k+m}} \right). \tag{29}$$

Given that $\mu_j = j\theta$, the ratio of ϕ_{k+m} and ϕ_{k+m+1} can be written as

$$\frac{\phi_{k+m}}{\phi_{k+m+1}} = \frac{e^{-\mu_{k+m}(\overline{\xi}-\overline{d})}}{e^{-\mu_{k+m+1}(\overline{\xi}-\overline{d})}} \cdot \frac{\mu_{k+m+1}}{\mu_{k+m}} \cdot \frac{\prod_{i=k,i\neq k+m}^{\tilde{k}} \frac{\mu_i}{\mu_i-\mu_{k+m}}}{\prod_{i=k,i\neq k+m+1}^{\tilde{k}} \frac{\mu_i}{\mu_i-\mu_{k+m+1}}} = (-1)\cdot e^{\theta(\overline{\xi}-\overline{d})}\left(\frac{k+m+1}{k+m}\right)^2 \frac{m+1}{D-(k+m)}, \tag{30}$$

which enables us to get

$$\begin{aligned}\frac{\phi_k}{\phi_{k+m}} = \frac{\phi_k}{\phi_{k+1}}\cdot\frac{\phi_{k+1}}{\phi_{k+2}}\cdots\frac{\phi_{k+m-2}}{\phi_{k+m-1}}\cdot\frac{\phi_{k+m-1}}{\phi_{k+m}} &= (-1)^m\cdot e^{m\theta(\overline{\xi}-\overline{d})}\left(1+\frac{m}{k}\right)^2\frac{m!(\tilde{k}-k-m)!}{(\tilde{k}-k)!}\\ &= (-1)^m\cdot e^{m\theta(\overline{\xi}-\overline{d})}\left(1+\frac{m}{k}\right)^2\frac{1}{\binom{\tilde{k}-k}{m}},\end{aligned} \tag{31}$$

expressing elements ϕ_{k+m} as the product of ϕ_k and a simple coefficient, finalizing the proof. ■

Proof of Proposition 4 **Part a (Quasi-convexity):** For given $(\overline{d},\tilde{k},\theta)$, the random variable (RV) $X(k)$ is given as the sum of independent exponential RV and can by written as

$$X(k) = \sum_{j=k}^{\tilde{k}} X_j,$$

with X_j as an exponentially distributed RV with rate $j\theta$. Note that $X(k+1)-X(k) = -X_k$, $X_j > 0$ for any j, and $X_j >_{st} X_{j+1}$ for any j (see also Shaked and Shanthikumar (1988)). Total costs can be written as a function of random variable $X(k)$ in the following way:

$$\overline{\xi} - X(k) + (1+c)[X(k) - (\overline{\xi}-\overline{d})]^+. \tag{32}$$

The n-step difference function at point k of the above cost term, which we denote by $\Delta W(k,n)$, is given by

$$\begin{aligned}&\Delta W(k,n)\\ &= \overline{\xi} - X(k+n) + (1+c)[X(k+n)-(\overline{\xi}-\overline{d})]^+ - \left(\overline{\xi} - X(k) + (1+c)[X(k)-(\overline{\xi}-\overline{d})]^+\right)\\ &= X(k) - X(k+n) + (1+c)\left([X(k+n)-(\overline{\xi}-\overline{d})]^+ - [X(k)-(\overline{\xi}-\overline{d})]^+\right)\\ &= \left(\sum_{j=0}^{n-1} X_{k+j}\right) + (1+c)\left(\left[X(k)-(\overline{\xi}-\overline{d}) - \left(\sum_{j=0}^{n-1} X_{k+j}\right)\right]^+ - [X(k)-(\overline{\xi}-\overline{d})]^+\right)\\ &= Y(k,n) + (1+c)\left([X(k)-(\overline{\xi}-\overline{d})-Y(k,n)]^+ - [X(k)-(\overline{\xi}-\overline{d})]^+\right),\end{aligned} \tag{33}$$

where we set $Y(k,n) =_{st} \left(\sum_{j=0}^{n-1} X_{k+j}\right)$ for notational convenience. Note that $\Delta W(k,0) = 0 + (1+c)\left([X(k)-(\overline{\xi}-\overline{d})-0]^+ - [X(k)-(\overline{\xi}-\overline{d})]^+\right) = 0$ and that $Y(k,n+1) >_{st} Y(k,n)$ for all k and n. Function $\Delta W(k,n)$ is stochastically convex in $Y(k,n)$ because it is composed of the sum of a linear term and a convex function $[\cdot]^+$.

Let us assume that total costs in Equation (32) stochastically (non-strictly) increase at some k', i.e., that $\Delta W(k',1) = \Delta W(k',1) - \Delta W(k',0) \geq_{st} 0$. Then, because function $\Delta W(k,n)$ is stochastically convex in $Y(k,n) = \left(\sum_{j=0}^{n-1} X_{k+j}\right)$ and this term is stochastically increasing in n, it holds that $\Delta W(k',n) \geq_{st} 0$ for all $n \geq 1$, implying that the total costs function in Equation (32) for no $k > k'$ drops below its value at k'. The immediate consequence is that it is non-decreasing after its first increase, which also holds for the expectation and establishes quasi-convexity of $\widetilde{W}^{\Omega^{k,\check{d}(\cdot)}}$ in k.

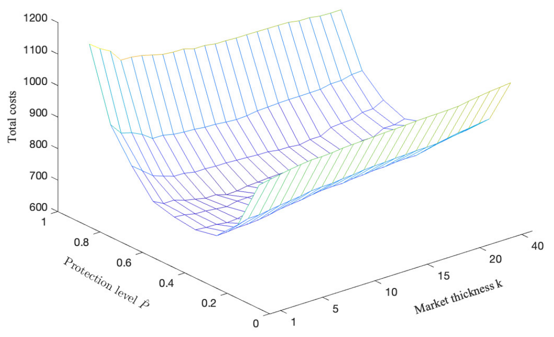
Part b (Submodularity): For given $(\overline{d}, \tilde{k}, \theta)$, as shown in the proof of part (a) of Proposition 4, term $\mathbb{E}[X(k) - (\overline{\xi} - \overline{d})]^+$ is non-increasing in k. Hence, for the difference function holds $(1+c)\mathbb{E}[X(k+1) - (\overline{\xi} - \overline{d})]^+ - (1+c)\mathbb{E}[X(k) - (\overline{\xi} - \overline{d})]^+ \leq 0$. Taking the derivative of the difference function in c shows the submodularity of the total costs in k and c. Moreover, given that set $k \in [1, \tilde{k}]$ is a sublattice and if we further assume that the domain of c is a poset, Theorem 6.1 in Topkis (1978) applies: Since $\widetilde{W}^{\Omega^{k,\check{d}(\cdot)}}$ is submodular in k and c, the optimal $k^* \in \kappa^*$ is ascending in parameter c. ■

7.4. Model and policy extensions

In this appendix, we study endogenous FPT, driver-type dependent compensation schemes, variable driver speed, clustering, driver abandonment, and a socially friendly adaptation of the BKT policy that limits the idle times of drivers.

7.4.1. Endogenous FPT ξ_i: In practice, the congestion level of a restaurant may affect food processing times and customer behavior. In the main model that we describe in Section 3, we assume that neither the distribution of the FPT nor the arrival rate of orders to a certain restaurant are affected endogenously by the system state. In this subsection, we extended our model to allow the FPT and the arrival rate of orders to be affected by the congestion level of the restaurant. If congestion is high, food processing may take longer. In addition, the customer who observes the congestion may cancel the order and search for alternatives. Therefore, we integrate the real-time congestion level of a restaurant in the endogenous FPT prediction. To this end, we modify the simulation procedure with the following changes: We introduce a state variable 'number of orders waiting-to-be-fetched' for each restaurant to indicate its real time congestion level, that we classify as low, medium or high for simplicity. For example, at 15 orders or above (3 orders or below), we define the congestion level as high (low). Then, we adjust the arrival rate of the restaurant and/or its FPT distribution to this congestion level. We implement this situation in the simulation by advancing the next order arrival for this restaurant by 5 minutes if the congestion is low, and by postponing it by 5 minutes if the congestion is high. In the same sense, we also stretch (advance) the FPT by 10% of its predicted value if the congestion of the restaurant is high (low).

The left-hand side of Figure 10 shows the costs of the BKT policy for different values of k and buffer level under the modified arrival rate, and the right-hand side of Figure 10 indicates the costs of the BKT policy for the case of modified FPT with $\breve{d}^* = 300$. In both cases, we observe the same quasi-convex structure that we observed in the original model. For the left-hand side, we obtain minimal cost $W^{\Omega^{k^*,\breve{d}^*,\hat{P}^*}} = 699$ at $k^* = 8$ and $\hat{P}^* = 0.6$, which is slightly higher (+10.3%) than the costs in the original model. For the right-hand side, we obtain minimal cost $W^{\Omega^{k^*,\breve{d}^*,\hat{P}^*}} = 684$ at $k^* = 5$ and $\hat{P}^* = 0.4$, which is +7.9% higher than the costs of the original model. It is intuitive that costs increase in these scenarios, as the congestion level introduces additional uncertainty to the model.



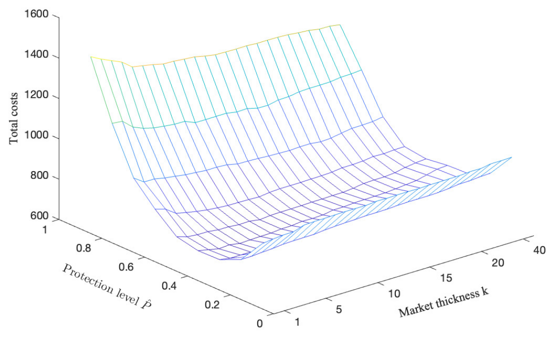


Figure 10 **Total costs under BKT policy as a function of thickening and protection levels for endogenously affected arrival rate (left-hand side) and FPT (right-hand side).**

7.4.2. Driver-type dependent compensation schemes: In practice, the unit fetch cost may depend on the type of driver (inhouse or crowdsourced) and the compensation scheme affect the matching decisions (*which type to match*). In this subsection, we extend the (B)KT policy to consider driver-type dependent compensation. To this end, we introduce two separate driver-reward functions $r_{in}(x)$ and $r_{cs}(x)$ for inhouse and crowdsourced drivers, respectively, with $x = d_{ij}(t_{ij}) + (\tau_i + \xi_i - t_{ij} - d_{ij}(t_{ij}))^+$. Reward function $r_{in}(\cdot)$ includes a fixed commission and a variable payment for each order; $r_{cs}(\cdot)$ only has a variable compensation for completed orders. Specifically, we set the fixed commission to 300, which is around $300/3600 \cdot 20 = 1.67$ yuan per order. We assume that the unit variable cost paid to a crowdsourced driver is +50% higher than that of an inhouse driver. We obtain $r_{in}(x) = 300 + x$ and $r_{cs}(x) = 1.5 \cdot x$.

We take the cost differences between both driver types into account in decision making by applying a transformation of the edge-expiration-timing of crowdsourced drivers. Given the reward functions, we redefine the edge expiration time of a crowdsourced driver to the time when the matching of an inhouse driver leads to the same cost as the matching of the crowdsourced driver at the edge expiration time, $\dot{t}_j$, i.e, $r_{in}(\dot{t}_j) = r_{cs}(t_j)$. We then adapt the BKT policy to take into account the re-defined edge times. For example, with our numbers, the redefined edge expiration time of a crowdsourced driver is $\dot{t}_j = r_{in}^{-1}[r_{cs}(t_j)] = 1.5t_j - 300$ if $t_j \geq 600$ (which ensures $r_{in}(600) = r_{cs}(600)$) otherwise, $\dot{t}_j = t_j$. The crowdsourced driver will then be checked whether to match at $\dot{t}_j$ time units before the end of the food processing. If matched, the order and the crowdsourced driver exit the system together. If not matched, the driver becomes inactive until t_j time units before the FPT, after which, she can be used for delayed matching and incur penalty cost. Note that even if matched at t_j, the order is also delayed but with zero penalty cost upon this moment.

Next, we test the adapted BKT policy for different values of variable commission of crowdsourced drivers and different values of fixed commission for inhouse drivers. In Table 6, we see that total costs increase in the variable commission of crowdsourced drivers. We also find that the share of matched inhouse drivers slightly increases, as the matching of crowdsourced drivers becomes more expensive.

Table 6 **Results of adapted BKT policy $W^{\Omega^{k^*,\breve{d}^*,\hat{P}^*}}$ by variable commission for crowdsourced drivers with $\hat{P}^* = 0.4, \breve{d}^* = 300$.**

Var. commission	1x	1.2x	1.4x	1.6x	1.8x	2.0x
Total costs $W^{\Omega^{k^*,\breve{d}^*,\hat{P}^*}}$	904	944	982	1031	1066	1110
Share of inhouse drivers (%)	68.13	68.27	68.35	68.40	68.67	68.68

The results in Table 7 show that total costs also increase in the fixed commission for inhouse drivers. This time, the share of matched inhouse drivers slightly decreases because of the increased cost for this driver type.

Table 7 **Results for adapted BKT policy $W^{\Omega^{k^*,\breve{d}^*,\hat{P}^*}}$ by fixed commission for inhouse drivers with $\hat{P}^* = 0.4, \breve{d}^* = 300$.**

Fixed commission	100	200	300	400	500	600
Total costs $W^{\Omega^{k^*,\breve{d}^*,\hat{P}^*}}$	765	836	904	972	1040	1108
Share of inhouse drivers (%)	68.50	68.13	68.13	68.13	68.13	68.13

7.4.3. Variable driver speed: Liu et al. (2021b) found that drivers may speed up to reach the restaurant on time and avoid overtime penalties. Next, we study how such behavior would affect our results by modifying our model: With probability p_{vd}, a delay event happens that threatens the timely arrival of a driver at the restaurant. In the simulation, if an order is matched after the edge expired, it is automatically treated as potentially delayed. In this case, the platform offers a fixed compensation of c_{up} if the driver speeds up, and the driver arrives on time with probability p_{vt}. In this case, the platform pays c_{up}. If the driver does not arrive on time, the compensation is not paid out.

We numerically analyze the results of this scenario. We set $\breve{d}^* = 300$, $p_{vd} = 0.05$, $c_{up} = 600$ (∼3 yuan), and $p_{vt} = 0.5$, and vary the probability values. The results are shown in Table 8. We find that the total costs are positively increasing in the delay probability p_{vd}, indicating the necessity for the planner to take into account potential delay events. We also find that total costs are decreasing in p_{vt}, indicating that less reliability leads to higher costs. We conclude that incentives to motivate drivers to speed up may reduce overtime penalties.

Table 8 Results for BKT policy $W^{\Omega^{k^*,\breve{d}^*,\hat{P}^*}}$ as a function of probability p_{vd} and probability p_{vt} with $\hat{P}^* = 0.4, \breve{d}^* = 300$.

Delay probability p_{vd}	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09	0.1
Total costs $W^{\Omega^{k^*,\breve{d}^*,\hat{P}^*}}$	638	642	645	650	654	657	659	662	664	668
On-time probability p_{vt}	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Total costs $W^{\Omega^{k^*,\breve{d}^*,\hat{P}^*}}$	669	665	662	657	654	652	649	646	642	638

7.4.4. Regional clustering: In Section 3, we defined the neighborhood of orders as the set of all available drivers. But Li and Netessine (2020) found that redundant market thickness may lead to search friction and consequently to a low matching rate. This motivated us to test the effect of our neighborhood assumption by analyzing *clustering*.

To this end, we divide the selected service region into n_c fixed *clusters*. Table 9 shows the results for $n_c \in \{1,2,4,8\}$ and $\breve{d} = 300$, where $n_c = 1$ is the model that we studied in the main part of the paper. We report the percentage gap against this model, $W^{\Omega^{k^*,\breve{d},\hat{P}^*}}(n_c = 2) - W^{\Omega^{k^*,\breve{d},\hat{P}^*}}(n_c = 1))/W^{\Omega^{k^*,\breve{d},\hat{P}^*}}(n_c = 1) \times 100$. The tests results indicate that clustering of a region may be beneficial, particular respecting to certain performance measurements. For example, for $n_c = 2$, both the expected delay per order V_Ω^d (-5%) and overall service level (+5%) have been improved, which might be due to reduced search friction, as discussed above. Moreover, if we continue to divide the region into more clusters ($n_c = 4$ and $n_c = 8$), the total costs increase because the average delay time V_Ω^w is seriously increased compared to the case $n_c = 2$. This reveals that, by further dividing the region into more clusters, the clustering restricts the availability of drivers and leads to a thin marketplace.

Intuitively, each subregion may have different optimal parameters for the (B)KT policy. We show that for case $n_c = 4$ in Figure 11, the quasi-convex structure of the total costs that we showed in Subsection 4.3 for the simplified model can also be observed for each subregion, but the optimal values of k^* and $\hat{P}^*$ differ between subregions (e.g., given $\breve{d} = 300$, $k^* = 4$ and $\hat{P}^* = 0.3$ for subregion 1 while $k^* = 2$ and $\hat{P}^* = 0.2$ for subregion 4).

Our analysis shows that clustering may lead to certain benefits (in terms of service level, i.e., matching rate) and optimal partitioning of clusters (e.g., Carlsson et al. 2021) remains an important research problem for on-time last-mile delivery.

Table 9 Percentage gap of total costs for $n_c \in \{1,2,4,8\}$ clusters.

	Total Costs	Drive-to-Shop	Delay	Service Level	Engagement Time	Engaged Drivers	Waiting Time	Idle Time	Net-income
Policies	$W^{\Omega^{k^*,\bar{d},\bar{P}^*}}$	V^p_Ω	V^d_Ω	V^l_Ω (%)	V^e_Ω	V^n_Ω	V^w_Ω	V^q_Ω	V^s_Ω
$n_c = 1$	634	110	39	64.42	1201	92	291	1522	8.7
$n_c = 2$	662	118	37	68.90	1242	95	324	1883	8.2
$n_c = 4$	735	112	58	62.68	1188	91	277	2661	6.3
$n_c = 8$	762	127	59	62.42	1208	93	281	1473	10.8

All time values in seconds.

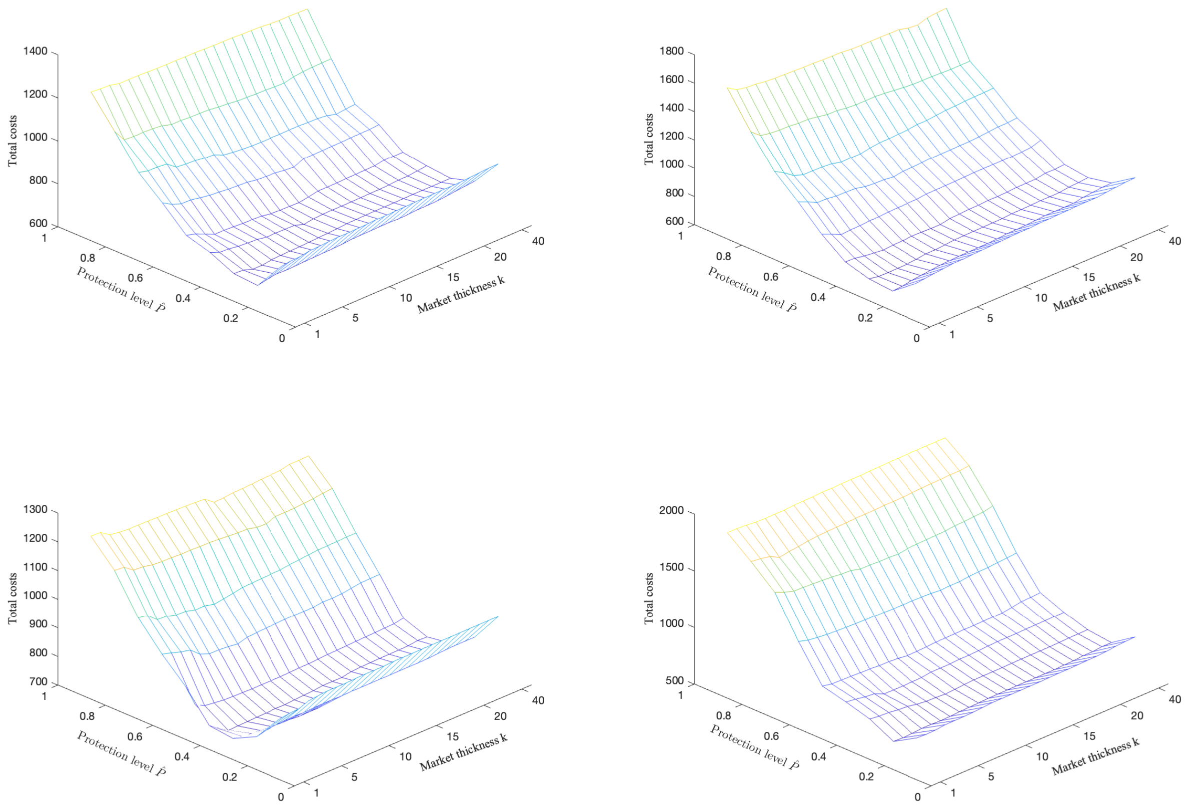


Figure 11 Total costs under BKT policy as a function of thickening and protection levels under $n_c = 4$ subregions.

7.4.5. Socially friendly BKT policy: The results of Section 5 indicate that the (B)KT policy effectively reduces the platform's cost by reducing the drivers total engagement time. However, the objectives of platform and of drivers are conflicting, and this effect leads to a decrease in expected net-income of the drivers, because of the longer (non-paid) idle time of drivers. In this subsection, we therefore propose a *socially friendly* adaptation of the (B)KT policy, by limiting the length of idle times for drivers.

If a driver exceeds a threshold y of idle time, it receives a compensation c_q and leaves the platform (e.g., transfers to another platform or exits for a break) without matching.

We perform a numerical experiment to study the impact of this adaptation of the BKT policy. We set the threshold $y \in \{30, 45, 60, 75, 90, 120, \infty\}$ minutes (the last case corresponds to the model in the main part of the paper) and the compensation to $c_q = 900 \cdot y/60$ (5 yuan per hour). Figure 12 shows total costs of the platform and driver net income as a function of y. Both, total costs as well as driver income, decrease with greater y, indicating the inherent conflict between the interests of the platform and those of the drivers (see also Meituan 2021). The platform has to give up on part of its profit to maintain a higher driver income. Our modified BKT policy allows the platform to reduce the burden of overly long idle times of drivers and, by appropriately setting threshold y, to find a balance for $W^{\Omega^{k^*,\check{d}^*,\hat{P}^*}}$ and V_s^{Ω} that corresponds to its driver availability and company values.

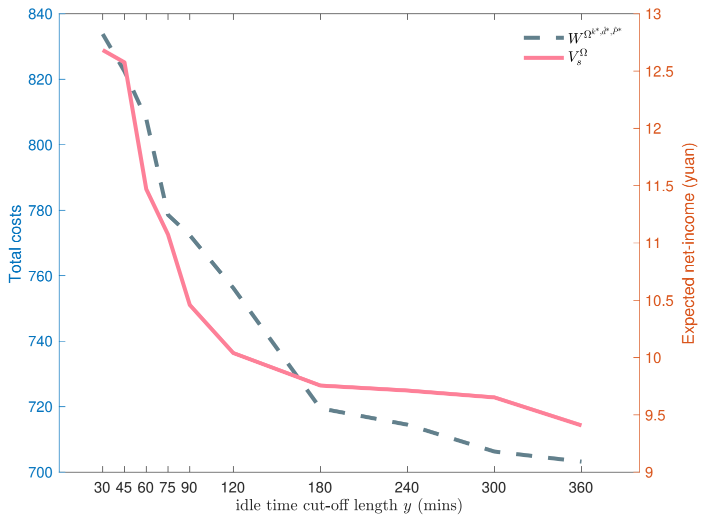


Figure 12 **Platform's total costs and driver's expected net-income as function of threshold y with $\hat{P}^* = 0.4, \check{d}^* = 300$.**

7.4.6. Abandonment rate of drivers: The assumption regarding driver abandonment is crucial for ensuring the long-term stability of the matching system. However, the abandonment rate, λ_a (i.e., the number of drivers abandoning per hour), also impacts the availability of circulating drivers within the system, consequently affecting both the driving time and total costs. We perform additional numerical tests to further investigate the influence of this parameter on the results.

In Table 10 we show the total costs per order and the average drive-to-shop time under different value of abandonment rate λ_a. A discernible trend becomes evident: as λ_a increases, both total costs, $W^{\Omega^{k^*,\check{d}^*,\hat{P}^*}}$, and drive-to-shop time, V_{Ω}^{p}, increase. Total costs increase significantly between $\lambda_a = 0$ and

Table 10 **Results for BKT policy $W^{\Omega^{k^*,\check{d}^*,\hat{P}^*}}$ as a function of abandonment rate λ_a.**

Abandonment rate λ_a	0	1/20	1/10	1/5	1/2	1	5	10	20
Total costs $W^{\Omega^{k^*,\check{d}^*,\hat{P}^*}}$	634	634	634	630	647	691	913	1017	1153
Drive-to-shop V^p_Ω	110	110	110	106	108	116	154	191	211

$\lambda_a = 20$, starting at 634 and ending at 1153 (+82%). This trend suggests that higher abandonment rates lead to higher costs (under the BKT policy), which is a consequence of the reduced availability of eligible drivers in the system. The evolution of the drive-to-shop time V^p_Ω appears to be more consistent than that of the total costs as λ_a grows.

7.5. Solving the Example 1 with KT policy

In this subsection, we apply the KT policy ($k = 1$ and $\check{d}(\cdot) = 2$) to Example 1 (see Section 4). In Figure 13, on the left-hand side, we observe that at $t = 1$, the connection between Driver A and Order 1 is about to expire. However, with the arrival of Driver B, we know that there are two compatible drivers for Order 1. Consequently, we do not match Driver A at this moment. At time $t = 2$, Driver B's availability expires, marking the last compatible driver for Order 1. Consequently, we match Order 1 with Driver 1 at this time. Simultaneously, Order 2 enters the system at time $t = 2$. At $t = 4$, the connection between Driver A and Order 2 is about to expire, leading us to match both. In this scenario, Driver B and Order 1 have exited the system, and we represent this in the right-hand side of Figure 13 (shadowed area). We conclude that the KT policy takes the same decisions as the optimal policy presented in Figure 3.

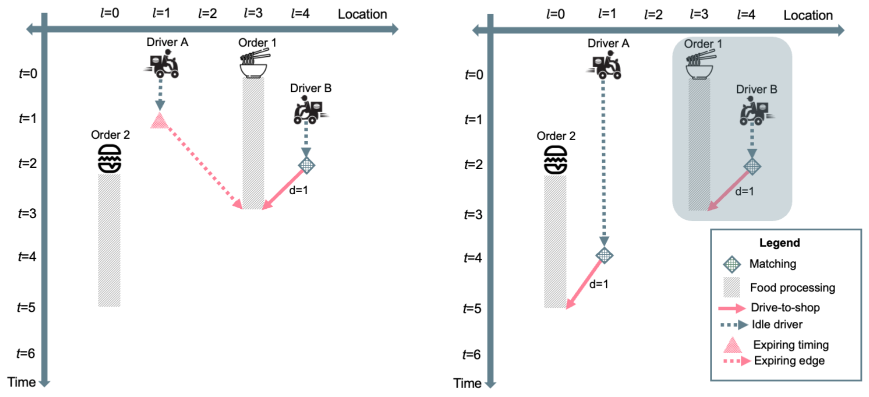


Figure 13 **Illustration of the KT policy applied to Example 1.**