Optimizing Task Generation and Assignment in Crowdpicking

Yasemin Ovalı
Department of Industrial Engineering, Koc University, Istanbul, Türkiye, yovali13@ku.edu.tr

Barış Yıldız
Department of Industrial Engineering, Koc University, Istanbul, Türkiye, byildiz@ku.edu.tr

Abstract: As omnichannel operations become increasingly important for meeting diverse customer expectations in retail, continuous innovation in service and business models is essential to maintain a competitive edge. While effective order fulfillment is key to omnichannel success, the manual picking process in physical stores, one of the major driver of fulfillment costs, still offers substantial opportunities for improvement. This paper focuses on the *crowdpicking* model as an innovative approach to manage online order picking operations in physical stores by leveraging existing in-store customers, offering a business model with considerable potential. We explore the real-time assignment of orders to in-store customers using ML to identify effective assignment policies. These policies are combined with a task-decomposition strategy to reduce picking costs and enhance crowdpicker participation as a key resource. The proposed crowdpicking model and its real-time management framework are tested on real-world data. Our results show that a well-managed crowdpicking system can lower order picking costs by more than 20% and provide actionable insights for managers in designing such systems.

Key words: order fulfillment, ship-from-store, crowdsourcing, machine learning, sequential decision-making.

This is the accepted version of the article accepted for publication in Production and Operations Management. The final version is available at https://doi.org/10.1177/10591478251377850.

This version is made available for non-commercial use only.

1 Introduction

Reshaping the retail landscape, omnichannel operations are expanding rapidly to meet diverse customer expectations. By integrating online and offline shopping experiences, omnichannel sales allow retailers to take unique advantages of physical and online presence. As omnichannel sales become the norm, traditional brick-and-mortar (B&M) retailers are increasingly adopting online channels, while digital-native brands are opening physical stores (Rigby 2011).

The grocery sector, which arguably faces some of the greatest challenges in conducting online operations, is no exception to this trend. As one of the most competitive sectors in retail, the grocery industry is under significant pressure to expand its online presence. In fact, e-commerce revenue in grocery is projected to surpass 257 billion U.S. dollars in 2024, accounting for approximately 25% of the total grocery

market revenue (Statista 2024). However, as Gallino and Moreno (2019) highlight, retailers with perishable products such as groceries face unique omnichannel fulfillment challenges, including heightened urgency, greater SKU variety, and highly variable labor demand. The wide variety of products in each order adds complexity to fulfillment, with the average e-grocery basket containing up to 50% more items than a typical non-food online order (Zatz et al. 2021). Tight time-frames for completing order fulfillment, such as same-day or even hourly deliveries, further amplify these challenges. To compound the difficulties, grocery retailers must manage these complex and costly fulfillment operations while operating on razor-thin profit margins (Asdemir et al. 2009), which are typically around 2% (The Economist 2023), substantially lower than the 5% margins observed in non-food retail sectors such as apparel.

As grocery retailers navigate the challenges of omnichannel retail, efficient order fulfillment becomes essential for maintaining competitiveness and achieving operational success. Among the various fulfillment strategies, ship-from-store operations have gained considerable popularity. This model offers several key advantages: it reduces the need for additional investment by leveraging existing store infrastructure, expands the online product assortment by utilizing in-store inventory, and enables faster delivery times due to store proximity to customers. However, balancing the needs of walk-in and online customers introduces significant complexities, particularly in order picking processes. Store layouts and product placements are primarily optimized to enhance the in-store shopping experience, often creating inefficiencies when repurposed for picking operations (Difrancesco et al. 2021). Also, while advanced picking technologies are commonly employed in warehouses or dark stores to enhance efficiency, their use in customer-facing retail environments is generally impractical, as they risk disrupting the shopper experience. Consequently, order picking in dual-purpose stores relies heavily on manual labor, leading to elevated staffing costs (Hübner et al. 2016). In fact, more than 60% of omnichannel retailers report increased operational complexity and fulfillment cost pressures when integrating stores into their fulfillment networks (Hübner et al. 2016). These challenges are further compounded by findings showing that in-store picking operations can be 30–40% less efficient than picking in dedicated centers (Seghezzi et al. 2022b). To exacerbate the issue, demand variability in grocery retail is substantial, fluctuating significantly throughout the week and even within daily operating hours leading low employee utilization during off-peak hours and further straining operational efficiency.

To address these challenges, crowdsourcing (CS) models are gaining attention in academia and practice as a viable alternative to use store employees for order fulfillment. Platforms like Instacart and GrubHub have popularized CS by connecting customers with personal shoppers who retrieve and deliver groceries from local stores. The primary advantage of CS is its scalability and flexibility, allowing retailers to meet fluctuating demand. However, these platforms commonly operate with a revenue-sharing model, and commission rates can significantly reduce the profit margins of groceries (Haddon and Jargon 2019).

As a response to these limitations, Dayarian and Pazour (2022) propose a novel CS model aimed at improving the efficiency of in-store picking—a critical yet often overlooked component of omnichannel

fulfillment. This is particularly important given that picking activities alone can represent over 55% of total operational costs related to labor and in-store handling (Wildt et al. 2024, Vazquez-Noguerol et al. 2022). Their approach suggests leveraging in-store customers as ad-hoc pickers, rather than relying on dedicated shoppers employed by CS platforms or store employees. This distinction is similar to a well-known differentiation in the transportation sector between dedicated crowd-shippers (independent contractors committed solely to delivery tasks) and occasional drivers, who opportunistically assist with deliveries during their planned personal trips (Yıldız 2021). Since occasional drivers would undertake their trips regardless of the deliveries, their earnings expectations for the additional service are significantly lower than those of dedicated crowd-shippers, who make trips specifically for deliveries. Similarly, in-store customers who would already do their shopping would have a lower marginal cost than dedicated pickers (Dayarian and Pazour 2022). However, assigning tasks to in-store customers under changing in-store and online store conditions remains a key challenge for such models. To address this, we propose key novelties to enhance both the acceptability and effectiveness of the suggested approach.

Firstly, we develop a real-time assignment policy aimed at minimizing picking costs, which can immediately determine which picking task (if any) the system should offer to a candidate crowdpicker (CP) when they express their willingness to participate. This is a critical capability to enhance CP participation as in the system, CPs are not contractors who can afford to wait for tasks. Requiring them to notify the store of their arrival in advance (extra time for the system) or making them wait upon arrival to learn their picking task can have a significant negative impact on participation. However, the uncertainty regarding future CP and order arrivals complicates the decision-making process, making it challenging to determine assignment decisions within a short time frame (i.e., in a few seconds). As we show in our numerical experiments, basic heuristics that can provide solutions quickly fail to provide high-quality solutions to this complex stochastic decision problem. On the other hand, more sophisticated optimization approaches that can effectively deal with uncertainties require too long computational times. We address this challenge with an innovative approach that avoids solving a complex optimization problem at each decision epoch. Instead, we leverage a Machine Learning (ML) model to identify cost-effective assignments for a CP in the given system conditions. To achieve this, we generate problem instances with historical data, find optimal matchings assuming perfect information about CP and order arrivals, and use these historical results to train the ML model. With the help of the ML model's predictions on the efficiency of a match, we determine assignment decisions that can achieve costs that are as good as the minimum costs one could get by deciding the matching decisions with perfect information. Increasing the significance of the methodological contribution of our study, the proposed solution framework is highly flexible and can be adapted to develop efficient system management policies across various applications that involve solving complex real-time matching problems.

In addition, to enhance CP participation and system efficiency we partition the orders into Assignment Units (AUs). Decomposing online orders into smaller units offers several benefits: (1) it helps find better

matches with shopping lists, minimizing deviations from in-store customers' intended shopping paths; (2) it reduces the likelihood of order collection failures. When tasks contain items from multiple aisles and different product categories, the risk of errors in picking and packing increases. By decomposition, users can pick fewer items, which not only decreases the error rate but also makes it easier to double-check the products; and (3) it increases the acceptance rate in CS systems, where simpler tasks tend to attract higher participation rates Kobren et al. (2015). In fact, the empirical analysis conducted by Dayarian and Pazour (2022) indicates that the least deviation from the shopping path and simplicity of tasks is critical for the majority of the respondents. Dividing online orders into smaller units not only enhances the CP experience, but also significantly reduces picking costs. Indeed, in our numerical experiments, we observed that the proposed decomposition approach led to a 40% reduction in total costs by decreasing the time-load of picking through better alignment of CPs' shopping lists and picking assignments.

Finally, to our knowledge, the proposed crowdpicking concept has not yet been tested with real data from a grocery store serving both online and offline customers. Our study contributes to addressing this critical gap by applying the crowdpicking model to actual dual-serving data.

The remainder of the paper is organized as follows. **Section 2** provides an in-depth review of the literature. **Section 3** outlines the problem context and presents a study conducted to assess its practical relevance. **Section 4** introduces the key notations and definitions used in the model and explains the details of the methodology developed to address the problem. **Section 5** presents the findings from empirical tests of our proposed approach, comparing it with other potential methods and discussing the implications for management. Finally, **Section 6** concludes with final remarks and a discussion of potential areas for future research.

2 Literature Review

The rise of store-based fulfillment has significantly reshaped the retail landscape. While this approach offers clear advantages such as reduced infrastructure investment, faster delivery, and improved utilization of in-store inventory, it also introduces substantial operational complexities. These challenges emerges from the dynamic nature of physical stores, where real-time customer traffic, spatial constraints, and demand volatility collectively challenge the consistency and efficiency of fulfillment operations (Milligan et al. 2025). To address such complexities, recent research has explored structured planning frameworks that integrate decisions across multiple functions and time horizons. For instance, Hübner et al. (2022) propose a hierarchical framework that expands planning strategies across multiple horizons and levels, including assortment management, order-to-fulfillment assignment, and replenishment decisions, while balancing practical decomposition with integrated decision-making across tasks.

Building on this need, a growing body of research has emerged on how to scale store-based fulfillment efficiently to address these challenges. Key concerns in store-based fulfillment include workforce adaptability and fulfillment costs. Unlike warehouses, stores often lack dedicated labor for fulfillment tasks, making

it challenging to balance picking efficiency with service quality—despite offering greater flexibility and shorter delivery times (Vazquez-Noguerol et al. 2022). Studies such as Difrancesco et al. (2021) show that batching orders can improve picker productivity, but may delay order readiness. Alternatively, assigning dedicated pickers can accelerate picking, yet significantly increases labor costs, highlighting the trade-off between efficiency and customer expectations. Also, Dethlefs et al. (2022) show that dynamically assigning online orders to stores can reduce lead times while increasing coordination and computational burden.

To address this, Bayram and Cesaret (2021) construct an intuitive heuristic policy that helps retailers make omnichannel fulfillment decisions while ensuring scalability. Additionally, several models have been proposed to optimize picker allocation and order assignment under uncertainty. For instance, Jiu (2022) develop a two-phase approach that combines anticipative replenishment with reactive allocation to address stochastic demand. Similarly, Baron et al. (2024) propose lightweight policies for order-store assignment under acceptance uncertainty. Recognizing the interdependence between inventory availability and fulfillment performance, Goedhart et al. (2023) introduce an integrated model that jointly optimizes replenishment and in-store fulfillment decisions in omnichannel settings. Finally, Mou (2022) emphasize the importance of jointly optimizing batching and assignment decisions in omnichannel supermarkets, accounting for workforce heterogeneity and store crowding.

Also, store congestion itself is a persistent barrier to scale in-store fulfillment. Conducting logistics operations in spaces shared with customers causes frequent disruptions. To mitigate these, several studies have explored spatial optimization strategies. For instance, Neves-Moreira and Amorim (2024) use a Markov Decision Process to route pickers away from congested aisles, while Pietri et al. (2021) propose proximity-based heuristics to reduce customer interference during picking. This has also led to interest in hybrid models, such as the backroom-and-aisle approach proposed by Seghezzi et al. (2022b), which uses mixed-integer programming to balance congestion and efficiency. In our study, we take a different perspective. Rather than improving order assignment and picking scheduling for dedicated workers to address the congestion issues, we explore the use of in-store customers as pickers.

While crowdsourcing (CS) has been extensively studied in the context of last-mile delivery (Mak 2018, Arslan et al. 2019), its potential within the store, particularly through in-store customers fulfilling online orders, remains largely underexplored. Although experts consider delivery and order picking to be equally critical, few studies have examined the role of CS in picking operations (Seghezzi et al. 2022a). In response to this gap, and inspired by the opportunistic labor paradigm (Yıldız 2021), the concept of crowdpicking redefines opportunistic fulfillment by utilizing the presence of customers within physical retail stores.

Dayarian and Pazour (2022) is the first study to consider crowdpicking. Their study provides empirical evidence supporting the practicality of the model in terms of attracting a sufficient number of CPs and develops an Approximate Dynamic Programming approach to solve the resulting stochastic task assignment problem. Consistent with our findings, their results show that a substantial share of orders can be fulfilled

through this approach, offering significant cost advantages by reducing the reliance on dedicated picking staff. Bhowmick et al. (2025) further explore this idea by examining a hybrid picking model in which in-store customers collaborate with autonomous mobile robots (AMRs) to fulfill online orders. Due to the stochastic nature of customer participation and synchronization with AMRs, they model the task assignment problem as a Markov Decision Process and propose a heuristic solution framework to dynamically update picking assignments. Similar to Dayarian and Pazour (2022), we consider a crowd-based picking system that focuses exclusively on human pickers rather than integrating autonomous agents. However, our work differs from theirs in three key dimensions.

First, we propose an order decomposition strategy that partitions customer orders into smaller units based on historical shopper behavior and system expectations, and we demonstrate that this strategy can yield significant efficiency gains. While splitting strategies to improve operational efficiency in store-based fulfillment have been explored (Difrancesco et al. 2021), such as zoned picking, pick-by-article methods, and batch-based release with sequential processing—these studies assume dedicated pickers without personal shopping lists. In contrast, our crowdpicking setting involves in-store customers with their own shopping needs. To address this, we introduce an iterative solution approach that actively shapes decomposition decisions to improve the assignment problem, which relies on these decomposed units as input.

Second, we develop a novel methodological framework for task allocation that directly maps system states to decisions, eliminating the need to solve an optimization problem at each decision point. Our problem falls within a broader class of dynamic assignment problems under uncertainty, often modeled using Markov Decision Processes (MDPs) (Goedhart et al. 2023) or Approximate Dynamic Programming (ADP) (Powell 2007), the latter also being employed by Dayarian and Pazour (2022). Sophisticated heuristic approaches—such as Tabu Search (Cordeau and Laporte 2003), Variable Neighborhood Search (Ropke and Pisinger 2006), and Rolling Horizon heuristics (Arslan et al. 2019)—have also been applied in similar task assignment settings. However, unlike these approaches, which either rely on low-resolution approximations to improve computational efficiency at the cost of potential suboptimality, or require solving large-scale optimization problems that hinder their applicability in real-time decision making contexts, our study introduces a novel machine learning-based framework that bypasses traditional optimization techniques by directly predicting decision variables from the current system state. To the best of our knowledge, this is the first approach in the literature to apply such a direct data-driven method to a sequential decision-making process in this context, offering a promising alternative to conventional methods.

Finally, our work complements Dayarian and Pazour (2022) by investigating the applicability of the proposed approach from the perspectives of businesses and using real-world online order and in-store customer data for a more realistic assessment of the potential benefits of the system.

3 Problem Description

3.1 Problem Setting

In this paper, we focus on the order picking operations of a single grocery store that utilizes in-store customers for collecting online orders. The system is designed based on the following set of principles.

- The store offers in-store shopping, online shopping with home delivery, and buy-online-pick-up-in-store (BOPIS) options.
- Each online order has an associated deadline for completing the picking operations. The time slot offerings and vehicle dispatching times that would determine the order picking deadlines are assumed to be predetermined by the store. The order needs to be collected before the deadline.
- To enhance the picking efficiency, online orders are divided into sub-orders referred to as *Assignment Units* (AU). To simplify the process of preparing orders for delivery and avoid mistakes, each AU can only contain items from a single order. AUs must comply with the weight and volume limits of the CPs.
- CPs are in-store customers who are willing to assist with the picking operations for online orders to earn some compensation. The store determines compensations based on the estimated extra time CP needs to spend collecting the assigned items (referred to as time-load here on) and their time valuations. Depending on their shopping profile (shelves typically visited for their own shopping) and their familiarity with the location of the products, the time load of a picking list (consisting of one or more items belonging to one online order) may vary for different CPs. Similarly, CPs may have different time valuations based on their attributes, such as age, gender, and shopping frequency. To account for this, CPs are segmented based on their personal attributes and/or past transaction data that align with the customer segmentation policies of the company from which we obtained the data. We apply the same per-minute compensation rates and time limits to all candidate CPs within the same customer segment. Similar to Dayarian and Pazour (2022), we assume that the store can accurately estimate the acceptable compensations for CPs (see EC.6 for robustness analysis) and that all assigned tasks are accepted, allowing us to focus on the core assignment model. We also analyze the system's sensitivity to changes in participation. As illustrated in Figure 5, relaxing this assumption does not significantly affect cost-effectiveness of the model as long as participation remains within reasonable bounds (e.g., around 50%).
- The system requests that customers volunteering as CPs inform the store of their shopping list (i.e., the departments they will be visiting for their own shopping) once they enter the store, to facilitate the suggestion of an optimal AU tailored to their route. This approach differs from the one described in Dayarian and Pazour (2022), where CPs are assumed to announce their arrival at the store 5-15 minutes beforehand.

The shopping list provided by the CPs is used in two key ways: first, to assign appropriate picking tasks that align with their intended route, and second, to determine their compensation based on the expected effort and efficiency. Importantly, while the system optimizes assignments based on the declared shopping

lists, in the real world CPs can pick additional items beyond their initially identified baskets if they choose to do so during their trip. The potential impact of such deviationsi analyzed in EC.1.

- Using in-store customers for order picking introduces an uncertainty in picking capacity. To ensure target service quality, the store also uses dedicated pickers (employees) when needed. These pickers can be positioned from the existing roles (e.g., cashier) or an entirely new role responsible for order fulfillment.

3.2 Potential Feasibility: Insights from Industry Experts

The absence of prior implementations of the crowdpicking model necessitates an in-depth assessment of its feasibility and potential adoption in practice. To validate the model from the perspective of in-store customers, we rely on the study conducted by Dayarian and Pazour (2022), where 131 customers were surveyed regarding their willingness to participate in a crowdpicking system.

To validate the model from a retailer's perspective, we conducted expert interviews with key stakeholders in the omnichannel retail sector. Specifically, we gathered insights from representatives of four major grocery chains in Turkey and a mid-size grocery in the UK. These interviews were conducted both via phone and in person, allowing for a more comprehensive discussion of the key aspects: the main motivations and barriers for adopting the system, the existence of similar implementations in the industry, and whether the experts had encountered this business model before.

We selected the experts to be as representative as possible, considering different company sizes and strategies. For the sake of confidentiality, we are unable to provide the company names. However, to better categorize the outputs of the survey, we explain the company features, focused markets, and sizes as follows:

- **Expert 1:** The retailer with the most stores in Turkey, operating 13,000+ stores across 81 provinces and over 900 of the country's 920 districts as a discount grocery chain, serving both urban and rural areas.
- **Expert 2:** One of the top 5 retail companies in Turkey, with over 3,500 stores. Among these, 220 stores are considered high-segment. The company operates across all 81 provinces.
- **Expert 3:** Another major player in the top 5 retail companies in Turkey, operating 1,000 stores in 56 provinces, primarily located in larger cities.
- **Expert 4:** A UK-based retailer with 5+ stores located in the cities of London and Manchester.
- **Expert 5:** A local chain in Turkey, ranked among the top 50 retail companies. It operates only in the Eastern Mediterranean region and does not have stores in major Turkish cities like Istanbul. The company has 30 stores in two provinces.

Firstly, we provided a detailed information about the system to ask their initial impressions. This question was designed to be open-ended, allowing participants to articulate their thoughts freely. Three out of five participants stated that the existing infrastructure is highly suitable and such gamification-based models are highly attractive for their customers. Notably, these three experts represented companies with the largest store networks (Expert 1, Expert 2 and Expert 3) indicating that the scalability and wide reach of the

infrastructure might be a factor in their positive evaluation. Their enthusiasm for the gamification model suggests that larger companies, with more extensive customer bases, are better positioned to leverage such solutions to engage their customers.

Additionally, two participants highlighted a potential secondary benefit, reducing personnel costs, particularly in the discount retail segment (Expert 1) cost was the major driver for the system. One participant expressed concerns that this model would only be viable for large-scale retail chains. This concern came from a UK retailer with relatively smaller operations, who stated that small companies would not be able to benefit from the economies of scale in such a model. In the second and third questions, participants identified the *most promising benefits* and *major barriers* of the system from predefined multiple-choice options, with an *Other* category for additional comments. Their responses are summarized in Tables 1 and 2.

Table 1 Perceived High-Potential Benefits of the System

Benefit	**E1**	**E2**	**E3**	**E4**	**E5**
Customer Loyalty		✓	✓		✓
Low Labor Cost	✓	✓		✓	
Workforce Management			✓		
Market Competitiveness	✓	✓	✓	✓	✓
Cross-Selling		✓			
Other					

Table 2 Perceived Major Barriers of the System

Barrier	**E1**	**E2**	**E3**	**E4**	**E5**
Customer Participation					
Increased Picking Time		✓		✓	✓
Perceived Complexity		✓	✓		
Picking Errors	✓	✓		✓	
Regulatory Compliance	✓		✓	✓	
Other		✓			

As seen in Table 1, all experts unanimously agreed that this system has the potential to create market competitiveness. The primary reason for this consensus is that such a system has not yet been implemented in retail stores, making it a novel innovation in the industry. Additionally, the majority of participants emphasized that cost advantages are among the most significant potential benefits of the system.

On the other hand, when examining the barriers presented in Table 2, the most critical challenge appears to be regulatory issues. This concern was raised by the UK company and two Turkish companies. This suggests that regulatory concerns may arise universally. However, the other two firms suggest that legal hurdles could be overcome by offering non-monetary compensation (e.g., coupons) instead of cash payments.

Additionally, the possibility of increased order picking time and the risk of order picking errors, are other challenges raised by experts. Order picking time, in particular, was highlighted as a major concern by experts from regular and high-end market segments, especially in urban areas where competition for meeting order deadlines is particularly intense. On the other hand, the discount market expert raised the issue of order picking errors, which could add additional costs to correct the order fulfillment process.

Also, one expert raised a concern regarding the potential misuse of the system, noting that CPs could intentionally manipulate their behavior to disrupt the operation (e.g., not completing tasks). Interestingly, insufficient customer participation and technological infrastructure were not perceived as major risks.

In the fourth question, participants were asked to evaluate the applicability of the system using a five-point Likert scale: 1 - Cannot be implemented, 2 - Difficult to implement, 3 - Moderately applicable, 4 - Largely applicable, and 5 - Immediately applicable. Two out of five experts stated that the system is largely applicable, while three participants considered it to be moderately applicable, indicating that the system is perceived as feasible, with a tendency towards high applicability.

Finally, participants were asked whether they had previously implemented a similar system and whether they had heard of the crowdpicking concept before. One expert from the discount market mentioned an application where customers report store cleanliness. This application, highlights an early example of customer-driven engagement in the retail environment. The expert noted that their company, with an extensive store network, faces significant challenges in maintaining consistent service quality across such a large scale. As a result, they are exploring customer-driven business models that can effectively leverage their existing customer base. Another expert (Expert 2) referred to a neighborhood-to-neighborhood delivery model. However, all participants stated that they had never heard of the crowdpicking concept before.

The interview findings indicate that the system is generally perceived as applicable and, as an innovative concept, it has been met with excitement by industry experts. However, its successful implementation would require addressing key challenges through careful design, operational adjustments, and stakeholder collaboration, which would also require further studies and analysis.

4 Model Formulation and Methodology

As depicted in Figure 1, the system management must make two critical decisions for effective real-time management of the order picking operations with CPs:

- How to divide orders into AUs - *Order Decomposition Problem (ODP)*
- How to assign these AUs to CPs or dedicated pickers - *Assignment Problem (AP)*

In the subsections that follow, we first present the model formulation, including the notations and a formal definition of the problem. We then describe the methodology developed to solve the *AP*, assuming that the Assignment Units (AUs) are predetermined. Finally, we introduce our approach for solving the *ODP*, which generates AUs in a way that aims to improve overall system efficiency.

4.1 Notation and Problem Definition

Let Θ be the set of online orders to serve in a day during the operation hours $T = [0, t_{max}]$. For an order $\theta \in \Theta$, t_θ indicates the order arrival time, d_θ is the picking deadline, and ℓ_θ indicates the list of products requested by the customer. Orders are known by the system only after they arrive. At a given time $t \in T$, the set of open orders, i.e., the set of arrived online orders that are waiting to be collected, is denoted by Θ_t. To simplify the exposition, we will first focus on the assignment problem considering a predetermined procedure to divide orders into AUs. Later, we will explain how to determine the AUs in conjunction with the assignment policy to maximize the efficiency.

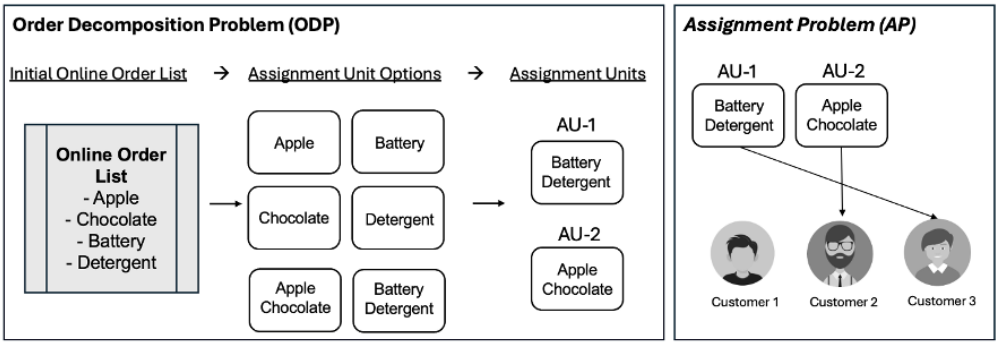


Figure 1 Order Assignment Process

An AU $\alpha \in \mathcal{B}_\theta$ is a subset of items from an order $\theta \in \Theta$ that is to be assigned to a CP for picking. The earliest time an AU α can be offered to a CP is shown by t_α, which is the arrival time of the order θ_α. Similarly, the deadline for an AU, d_α, is the latest time that the order θ_α must be collected for a timely delivery. The weight and the volume of an AU are defined as w_α, and v_α, respectively. Maximum acceptable volume and weight limit for picking are denoted as $\bar{W}$, and $\bar{V}$.

Let $\mathcal{K} = \{1, \ldots, K\}$ be the set of CPs visiting the store during the day, and M be the set of customer segments they fall into. Similar to the orders, the system becomes aware of a CP at the time it enters the store and declares their interest in assisting with the order picking operations. For a CP $k \in K$, t_k indicates the arrival time and ℓ_k is the set of products in their own shopping list. The customer's maximum acceptable time-load is denoted by $\bar{\tau}_m$ where CP k belongs to the segment m. For a customer $k \in \mathcal{K}$, the time it takes to collect the products in a shopping list ℓ is denoted by $\tau(\ell, k)$. For the sake of simplicity and to be consistent with the literature, in this study, we determine the time needed to collect items in a list by a customer by considering the traveling salesman tour with the minimum length (in time) that visits all the shelves that include the shopping items. However, we want to note that our methodology does not rely on this assumption. If needed, more sophisticated approaches that consider a wider range of parameters, such as familiarity with the locations of the products and personal habits regarding the order the shelves/product departments visited, can be adopted without any significant change in the methodology we develop.

For a CP k and an AU α, we define the time load δ_α^k as the difference between the minimum time to collect the items in the combined list $\ell_\alpha^k = \ell^k \cup \alpha$ versus the minimum time to collect only the items in the shopping list ℓ^k, i.e., $\delta_\alpha^k = \tau(\ell_\alpha^k, k) - \tau(\ell_k, k)$. For a CP segment $m \in M$, the per-minute earning expectation is denoted by c_m, and the cost of an AU α, for a CP k, is defined as $c_\alpha^k = c_m \delta_\alpha^k$. We assume that the dedicated pickers are always available and the cost of picking an AU α with a dedicated picker is denoted by $\bar{c}_\alpha$. When an AU α is given to a CP, the list of the open items (items waiting to be collected) of the order θ is updated by removing those that are included in the AU from $\mathcal{B}_\theta$. The order θ is removed from the list of open orders if all its items are collected or assigned to a CP. For a set orders $\bar{\Theta}$, and AU $\alpha \subseteq \hat{B}(\Theta)$, $\hat{B}(\Theta)$ is the set of AU where store created for a given set of open orders Θ_t.

We define set V to enumerate all valid CP-AU pairings. In practical terms, an AU cannot be assigned to a CP if it has not yet arrived or has already been delivered prior to the CP's arrival and if the volume, weight, or time load exceeds the CP's limitations. In other words, for a feasible CP $k \in \mathcal{K}$ from segment m and AU $\alpha \in \hat{B}(\Theta)$ pair, the conditions $d_\alpha \geq t_k \geq t_\alpha$, $w_\alpha \leq \bar{W}$, $v_\alpha \leq \bar{V}$, and $\delta_\alpha^k \leq \bar{\tau}_m$ must be satisfied. It is important to note that dedicated pickers are assumed to be available at all times, ensuring that the set V always includes these pickers, each with unlimited time capacity.

Utilizing the framework proposed by Powell (2019), we define the Assignment Problem (AP) as a sequential decision-making problem, where assignments are decided upon the arrival of each CP to the system.

- **State variable:** At a decision epoch t_k, $k \in \mathcal{K}$, we denote the system state as $S_k = \Theta_{t_k}$.
- **Decision:** At a state S_k, for the arriving CP k, the grocery decides the assignment $\alpha_k \subseteq B(\Theta_k)$, where $\alpha_k = \emptyset$ implies no assignment.
- **Policy function** At the core of the decision-making process, we define a policy function, $X^\pi(S_k) = \alpha_k$ that takes the state knowledge S_k to provide the assignment decision $\alpha_k \subseteq B(\Theta_k)$ for the CP $k \in \mathcal{K}$.
- **Transition function:** $\mathcal{S}(.)$ is the transition function that takes the current system state S_k and the decision α_k to produce the next state S_{k+1}. Let $\Theta_{k,k+1}$ be the set of grocery orders that arrive in the time interval $[t_k, t_{k+1}]$, i.e., $\Theta_{k,k+1} = \{\theta \in \Theta : t_\theta \in [t_k, t_{k+1}]\}$. Similarly, we define $\bar{\Theta}_{k,k+1} = \{\theta \in \Theta_k : d_\theta \in [t_k, t_{k+1}]\}$ as the set of open orders at time t_k whose picking deadline arrives until the next decision epoch t_{k+1}. We define $\mathcal{S}(S_k, \alpha_k) = \left(\Theta_k(\alpha_k) \cup \Theta_{k,k+1}\right) \setminus \bar{\Theta}_{k,k+1}$.
- **Cost function:** The single period cost function $C(.)$ takes the state S_k and the decision α_k and returns the summation of the CP compensation for the decided assignment and the cost of the dedicated pickers to collect the open orders whose picking deadlines comes until the next decision epoch (t_{k+1}). More precisely, we define $C(S_k, \alpha_k) = c_{\alpha_k}^k + \sum_{\theta \in \bar{\Theta}_{k,k+1}} \bar{c}_\alpha$. For easy reference, we present the outline of the basic notation in Table 3.

We define the Assignment Problem (AP) as finding the policy that minimizes the expected picking costs.

$$\min_\pi \mathbb{E}\left[\sum_{k=1}^{K} C(S_k, X^\pi(S_k) \mid S_0\right]. \tag{4.1}$$

4.2 Learning Based Assignment Policy

The primary challenge for solving AP lies in the uncertainty of future order and customer arrivals and the potential risk of missing opportunities to pair an AU with a CP who would have a better match, hence requiring a smaller compensation to collect it. If we had the foresight to know which customers and orders would arrive throughout the entire day, then the assignment decisions would transform into a basic matching problem. Solving this matching problem, we could offer the best CP-AU pairings that minimize the

Table 3 Outline of the Notation

Sets	
Θ	Set of online orders to serve
Θ_t	Set of open orders at a given time $t \in T$
T	Operation hours of the store
$\hat{B}(\Theta)$	Set of AUs store created for a given set of open orders Θ_t
$\mathcal{B}_\Theta$	Set of all possible AU combinations for a given set of open orders Θ_t
K	Set of CPs
K_θ	Set of CPs that are available for order $\theta \in \Theta$
M	Set of customer segments
I	Set of items available in the store
V	Set of valid CP-AU pairs
S_k	The state of the system at the decision epoch $k \in K$
$\Theta_{k,k+1}$	Set of grocery orders that arrive in the time interval $[t_k, t_{k+1}]$
$\bar{\Theta}_{k,k+1}$	Set of open orders at time t_k whose picking deadline arrives until the next decision epoch t_{k+1}
Functions	
$X^\pi(S_k)$	Policy function that takes the state knowledge S_k to provide the assignment decision α_k $k \in K$
$\mathcal{S}(S_k, \alpha_k)$	Transition function that takes current system state S_k and decision α_k to produce the next state S_{k+1}
$\mathcal{C}(S_k, \alpha_k)$	Cost function that takes the state S_k and the decision α_k and returns the cost
Parameters	
t_θ	Arrival time of order $\theta \in \Theta$
t_k	Arrival time of customer $k \in K$
d_θ	Dealine of order $\theta \in \Theta$
ℓ_θ	List of products requested by the customer in order $\theta \in \Theta$
t_α	The earliest time an AU α can be offered to a CP
d_α	Deadline for AU α
c_α^k	Cost of fulfill AU α by customer k $\in$ K
δ_α^k	Time load needed to fulfill AU α by customer k $\in$ K
c_m	Per-minute expectation for segment m $\in$ M
$\bar{c}_\alpha$	Cost of AU $\alpha \in A$ to a dedicated picker
$\bar{\tau}_m$	Maximum acceptable time-load for picking for CP $m \in M$

costs associated with order collecting. Building on this observation, we develop a novel Learning-based Assignment Policy (LAP) for solving the AP.

In LAP, we first generate problem instances from historical data, consisting of different order and CP arrivals, characterized by the pair of sets (θ, K). We solve the resultant matching problem from the perspective of a *Clairvoyant* decision-maker who possesses perfect foresight regarding CP and order arrivals, i.e., the sets θ and K are fully known to the optimizer. After solving the matching problem for a problem instance, we use the solution for labeling possible matchings of CPs with the AUs. For all possible matchings, those used in the optimal solution are labeled with 1 and others are labeled with 0. We refer to this initial phase as *Data Farming*.

In this step, the training data is constructed in a novel way by utilizing a hindsight-based optimization approach. The flow of the training data generation process is presented in Figure 2. The process starts with real-world in-store and online arrival data, followed by the instance generation, clairvoyant optimization, optimal matching solution, labeling of matchings, and the creation of the final training dataset.

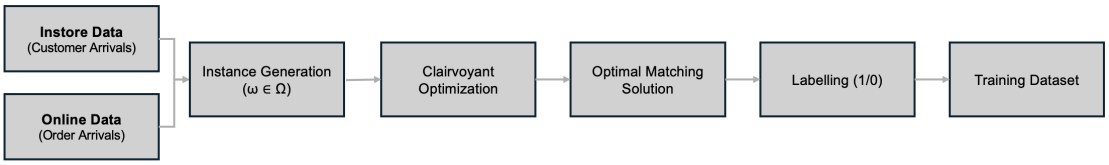


Figure 2 Training Data Generation Flow

After completing the data farming, we utilize the generated data to train a ML model, which is designed to predict the likelihood of a clairvoyant decision-maker matching a given AU to a specific CP. Finally, we apply a decision rule that combines the ML model's output with the current state information to make assignments to incoming CPs in real time. Next, we explain the details of each step.

4.2.1 Data farming

In the first stage of the data farming process, we derive problem instances from historical data. Let Ω be the set of these problem instances. For each instance, $\omega \in \Omega$, the corresponding orders and CPs are denoted by Θ^ω and K^ω, respectively.

For a problem instance $\omega \in \Omega$, let $y^\omega_{\alpha k}$ be the binary variable that indicates whether an AU $\alpha \in B(\Theta^\omega)$ is assigned to a CP $c \in K^\omega_\theta$. We use the following well-known matching model to determine the best assignment decisions with perfect foresight ($\mathcal{MP}$).

$$\text{Minimize:} \quad \sum_{(\alpha,k)\in V} \delta^k_\alpha \cdot c_k \cdot y_{\alpha k} \tag{4.2}$$

$$\text{Subject to:} \quad \sum_{(\alpha,k)\in V} y_{\alpha k} = 1 \qquad \forall \alpha \in B(\bar{\Theta}) \tag{4.3}$$

$$\sum_{\alpha\in\hat{B}(\Theta),(\alpha,k)\in V} y_{\alpha k} \leq 1 \qquad \forall k \in K_\theta \tag{4.4}$$

$$y_{\alpha k} \in \{0,1\} \qquad \forall k \in K_\theta, \alpha \in \hat{B}(\Theta) \text{ and } (\alpha,k) \in V, \tag{4.5}$$

The objective (4.2) is minimizing the total cost of order picking, including the payments to the CPs and the cost of using dedicated pickers. Recall that we define, δ^k_α as the time needed to collect AU $\alpha \in B(\bar{\Theta})$ by CP $k \in K_\theta$ and the time valuation of the CP (per minute) is given in c_k. We include the employee costs by adding a dedicated picker $\bar{k}$ to K_θ, where the cost of picking an AU α by the employee is set to $\bar{c}_\alpha = \delta^k_\alpha c_k$. Constraint (4.3) ensure that all AUs are allocated to a picker (CP or dedicated). We limit the number of AUs that a single CP can take to one in constraint (4.4). Finally, (4.5) indicates variable domain restrictions.

We use this matching model for two purposes. First, we solve this model to label the data we use to train and test the ML models. Second, we use its solutions as a benchmark (lower bound) to evaluate the quality of the assignments generated by the proposed learning-based policy as discussed in detail in Section 5.3.1.

4.2.2 Machine Learning (ML) Model

In this section, we detail the ML model developed for the study. While the ML models applied in this study are based on standard supervised learning techniques, their contribution lies in how they are strategically integrated into a sequential decision-making framework. Rather than solving the original optimization problem in real time, we train ML models on hindsight-optimal decisions generated by a clairvoyant policy. By learning from the outcomes of these fully informed solutions, the ML models approximate optimal decision-making in environments characterized by complex, sequential dynamics and inherently uncertain or incomplete information. This approach introduces a novel perspective, positioning ML not as a predictive tool but as a real-time decision-making engine.

Initially, we discuss the preprocessing methods we used to address the imbalance in the training data. Then, we present an overview of the features used in the model. We then discuss the criteria we considered to decide which ML algorithm to use, presenting the prediction performance of various classification methods. We conclude this section by investigating the feature importance scores in the final model to gain insights into how it captures the complex problem dynamics to determine good CP-AU matchings.

Data Preprocessing

In classification problems, one common challenge is the imbalance in class distribution (Tanha et al. 2020). This imbalance occurs when the instances of one class (the majority class) significantly outnumber those of the other class (the minority class). For example, in our training dataset, 'non-assigned' data instances outnumber 'assigned' instances because only a single CP-AU pair is selected in the optimal solution (labeled as '1'), while all other eligible pairs were labeled as '0', even in cases where multiple pairs had similar feature profiles. Consequently, the number of '1' values, representing assigned CP-AU pairs, equals the total number of AUs, whereas the '0' values, representing unassigned pairs, were disproportionately higher. This imbalance skews the model's ability to accurately generate classification scores. To address this issue, we test the following approaches:

- *Under Sampling:* We balance the dataset by randomly removing instances from the majority class (Hou et al. 2019), which is labeled as '0' in our case.
- *Relabeling:* As a symmetric approach to undersampling SMOTE is a widely-used class-balancing technique in ML that increases the number of minority class instances by generating synthetic samples (Mallah 2023). Similarly, we apply a relabeling approach to increase the proportion of the underrepresented class. However, instead of creating synthetic samples, we modify the labels of some of the data points in the majority class in the training data if there is another CP-AU pair that has very similar characteristics but is labeled as '1' (i.e., the optimizer had chosen to include this pair in the solution). This way we aim to discover whether there might exist an alternative solution in which this unused matching would be used. Specifically, for each CP-AU pair labeled as '1', we identify pairs that are labeled as '0' but have a lower or equal assignment cost, and fall within a similar 'time to deadline' category (e.g. urgent, moderate, low).

Feature Engineering

We build a machine-learning model using the 18 features, list of features and explanation can be found in EC.3, which we select to provide comprehensive coverage of all important factors that affect assignment decisions. These features can be classified into four main groups.

- *CP Characteristics:* Relevant attributes of candidate CPs—such as arrival time and segment—that are associated with assignment outcomes.
- *AU Priorities:* This group includes features related to the urgency of picking the items in the AU.
- *CP-AU Relations:* Features in this group focus on the interactions between AUs and CPs. Specifically, these features denote the time load of the AU for the CP, the associated cost that would be incurred by the system with this assignment, and the comparison of this cost with the cost of alternative assignments.
- *Operational Load:* This group consists of features that are related to the operational load of the system, which varies significantly throughout the day. These fluctuations can be critical in deciding assignments, necessitating adjustments based on real-time physical and online activity levels. For example, during peak in-store customer periods, it may be strategically advantageous to delay assigning an AU to an incoming CP, anticipating that a more favorable alternative could become available before the picking deadline. Conversely, when in-store activity is low but online demand is high, it may be necessary to assign tasks to CPs with higher-than-ideal costs to ensure timely order fulfillment and maintain service levels.

Model Selection

We selected the `XGBClassifier` (Chen and Guestrin 2016) as our ML model. This decision is based on performance comparisons for various algorithms, using the 'LazyPredict' library (Pandala 2024). Our comparison includes sophisticated models such as quadratic discriminant analysis, as well as simpler models like logistic regression. In these analyses, we use 80% of our data for training and the remaining 20% for testing. The performance of the models on the test set is reported in Table 4. As shown in the table, the `XGBClassifier` outperformed other models. Additionally, the `XGBClassifier`'s robustness in handling large datasets, scalability at managing a diverse set of features, and providing feature importance scores to get further insights contributed to our decision to proceed with this model.

The results for under-sampling techniques and the as-is training data are included in EC.2. As shown applying preprocessing techniques enhanced model performance from 85% to 99%. Among the tested methods, the relabeling technique outperformed both the as-is baseline and the under-sampling approach. Therefore, we proceed with the relabeling method for handling class imbalance.

Feature Importances

In this section, we analyzed the importance of different features using various measures, including feature importance scores and SHAP (SHapley Additive exPlanations) values. Figure 3 illustrates the results of our feature importance analysis. Specifically, Figure 3a provides a quantitative evaluation of each feature's

Table 4 Comparison of Different Classification Algorithms with Relabeling preprocessing

Model	Balanced Accuracy (ROC-AUC)	F1 Score	Precision	Recall
XGBClassifier	0,9939	0,9937	0,9936	0,9939
LGBMClassifier	0,9926	0,9924	0,9922	0,9926
RandomForestClassifier	0,9894	0,9892	0,9891	0,9894
BaggingClassifier	0,9858	0,9856	0,9854	0,9858
ExtraTreesClassifier	0,9850	0,9847	0,9845	0,9850
DecisionTreeClassifier	0,9829	0,9829	0,9829	0,9829
LabelPropagation	0,9674	0,9672	0,9670	0,9674
LabelSpreading	0,9665	0,9663	0,9661	0,9665
ExtraTreeClassifier	0,9529	0,9528	0,9526	0,9529
KNeighborsClassifier	0,9520	0,9519	0,9517	0,9520
SVC	0,9388	0,9384	0,9382	0,9388
AdaBoostClassifier	0,8855	0,8864	0,8883	0,8855
LinearSVC	0,8724	0,8734	0,8752	0,8724

impact on the model's prediction process and Figure 3b offers insights into how individual features influence the model's predictions. SHAP values explain the effect of each feature on a specific prediction relative to a baseline prediction, quantitatively capturing each feature's contribution (Lundberg et al. 2019). Each dot in the plot represents the SHAP value for a feature corresponding to an individual prediction. The red dots indicate higher feature values, while blue denotes lower values. For instance, blue dots positioned on the right side of the *cost* feature suggest that lower costs are associated with a positive impact on the model's predictions. So, if the model is predicting the likelihood of an option being chosen, lower costs are favored.



(a) Feature Importances

(b) SHAP Values

Figure 3 Feature Importance Analysis Graphs

As we see in Figure 3a, *Cost* emerged as the most significant feature in our model. This observation is in line with the structure of our assignment model, which prioritizes cost minimization. This validates our expectations and highlights the model's alignment with our cost-efficiency objectives. Given that the cost of an assignment often correlates with the additional time required for collection, it's probable that the influence of *Cost* on the model's predictions is prominent over the *Time Load* feature.

Cost was followed by the features *Orders up to now* and *Time to Deadline*. While these two features display similar levels of importance in the feature importance graph (a), their impact on the predicted

values varies slightly, as observed in Figure 3b. The broad distribution of *Time to Deadline* across the graph indicates that it can significantly influence the predicted value both positively and negatively. This variation suggests that the model treats short deadlines differently compared to longer ones. *Time to Deadline* is a aggregated measure that combines *Deadline Period* and *Time Period*. The integration of these variables into a single measure of urgency may explain why *Deadline Period* and *Time Period* individually appear to be less significant in the model. In contrast with the *Time to Deadline* feature *Time in the System* reflects the duration since an order was placed, and in the importance graph, it ranks moderately. In Figure 3b, this feature shows both positive and negative impacts, suggesting that the longer an order has been in the system, the more urgent its delivery may become, which can either positively or negatively affect the model's output. Also, *Deadline urgency* acts as the categorical counterpart to the *Time to Deadline* feature and is the inverse of it. In this feature, urgent orders are indicated as '3', while extended orders are indicated as '1'. Therefore, in Figure 3b, the variations of *Time to Deadline* and *Deadline urgency* are opposite. Similar to *Deadline Urgency*, the *Picking Lead Time* feature assesses the total time required to pick an order and holds almost equal importance to the *Deadline Urgency* feature.

On the other hand, the features *Customers up to now* and *Orders up to now* are intended to measure in-store and online store workload. Interestingly, both features exhibit a positive correlation with the outcome. This contrasts with our expectations for *Customers up to now* where an increase in customer numbers was anticipated to make the model's predictions more selective. Nonetheless, the observed trend may be attributed to a strong correlation between *Customers up to now* and *Orders up to now*. It is hypothesized that higher number of orders within a day leads to a higher likelihood of these orders being assigned to CPs. From the model's perspective, assigning orders to CPs is often more advantageous than dedicated pickers. Therefore, the model strategically favors CPs to reduce the proportion of orders handled by dedicated pickers. This justifies our expectation of a positive relationship for *Orders up to now* with the outcome.

We also incorporate the day of the week into the model using a numerical representation, where '0' corresponds to Monday and '6' represents Sunday. The positive impact of the *Day* feature within the model indicates that as the week progresses towards the weekend, the probability of assignment increases. This suggests a trend where assignments are more likely to occur as we approach the weekend. This result aligned with the *Orders up to now* and *Customers up to now* as store workload increases towards the weekend. Similarly, we've designed the *Time Period* feature with numeric values representing different times of the day: '1' for morning, '2' for noon, and '3' for evening. This feature aims to capture variations in in-store customers and order volumes throughout the day. However, it has shown to be less significant in influencing the model's results, likely due to varying periods of deadlines for online orders.

The features generated to compare different AU options (*Difference to Cheapest*, *Cheapest*) have a slight negative correlation between the prediction value, which is consistent with the impact observed with the impact of *Cost* feature. *Segment Code* has no impact on the model's predictive power, indicating that our

model operates independently of the segment. Instead, it prioritizes *Cost* and *Time to Deadline*, focusing on time efficiency and financial factors over customer segmentation.

Finally, we conducted feature selection based on importance scores. Certain features, such as *Remaining Orders*, *Remaining Earlier Orders*, *Time Period*, *Order Placement Period*, and *Deadline Period*, were excluded to eliminate non-correlated features from the model to minimize noise and over-fitting.

4.2.3 Incorporating ML to Assignment Decisions

Building on the ML results, we propose three strategies for real-time CP–AU assignment.

- **Best Score (BS):** A straightforward method for using ML predictions is to directly utilize the classification scores. Recall that, our classification model produces a score between 0 and 1, where values closer to 1 indicate a stronger likelihood that the pair is a *good* match. Therefore, in this strategy, when a new CP arrives, we evaluate the classification scores for all feasible AUs and allocate the CP to the AU with the highest score. However, this assignment only occurs if the score exceeds a predefined threshold, ensuring that a CP is not assigned to an AU if all scores are low, which indicates that assigning an AU to the incoming CP may not be a good decision. To determine the threshold, we generate a set of training problem instances (CP and order arrivals) as described in Section 4.2.1 and run a discrete event simulation (see EC.5) on these instances for alternative threshold values ranging from 0 to 1, increasing by 0.01 increments. We then choose the threshold value that gives the smallest total picking cost.
- **Score Ordering (SO):** Another approach is to treat classification scores as a proxy for the probability that a clairvoyant decision-maker, would choose a particular assignment. In this strategy, we begin by ranking the CP-AU pairs based on their classification scores, from highest to lowest. Starting with the first option, we randomly sample a number between zero and one. If the classification score is greater than the generated number, we assign the CP to the that AU. If not, we move to the next option and repeat the process until an assignment is made or the list is exhausted.
- **Score and Urgency (SU):** In this strategy, we consider both the ML scores and the remaining time before deadline (urgency). Similar to the BS approach, we establish a threshold for the ML scores to allow a match. However, unlike the BS approach, instead of selecting the AU with the highest score, we prioritize the AU with the least remaining time until the picking deadline, i.e., the one with the highest urgency. The method for determining the threshold follows the same procedure outlined earlier for the BS approach.

4.2.4 Benchmarks

In our numerical experiments in Section 5 we compare the performances of the ML-based policies described in the previous part with several benchmarks.

The first benchmark is the lower bound provided by optimal matching decisions from the clairvoyant model, assuming perfect information regarding the CP and order arrivals. We compare our proposed solution method with other practical benchmarks by how close they get to this theoretical bound.

As the second benchmark, we consider the Approximate Dynamic Programming (ADP) approach proposed by Dayarian and Pazour (2022), which represents the state-of-the-art for solving the crowdpicking assignment problem. Unlike our setting, their model assumes that customers' shopping lists are not fully known to the grocer, and assignment costs (based on time load) are estimated from the customer's five most recent transactions. For a fair comparison, we implemented the ADP method under the assumption that assignment costs are known precisely, consistent with our setting in which customers declare their shopping lists upon arrival and indicate their willingness to take on picking tasks. An extended analysis that incorporates cost uncertainty in both models is provided in EC.6.

Finally, to assess the added value of incorporating advanced solution approaches such as ADP and our proposed ML-based policy, we also evaluate two simple heuristics: (1) assigning the CP to the AU with the lowest cost, referred to as **Cheapest Assignment** (CA), and (2) assigning the CP to the AU with the most urgent picking task, referred to as **Urgency Assignment** (UA).

4.3 Order Decomposition Problem

In the previous section, we assumed that the AUs are already determined. Now, we introduce the decision-making process for dividing orders into AUs. We determine the order decomposition policy in three steps. Firstly, all possible AU combinations are enumerated for a given order list Θ. During this step, respective limitations are considered to determine *feasible* AUs that respect the time, weight, or volume limits. We denote the set of all feasible AUs by $\mathcal{B}_\Theta$. In the second step, we determine the expected cost of serving an AU by considering two parameters: (1) the probability that the created AU will be served by a CP from customer segment $m \in M$, denoted by ρ^m; and (2) the expected cost of assigning the created AU α to a CP from customer segment $m \in M$, denoted by c_α^m. The expected cost for serving an AU a is calculated as $\hat{c}_\alpha = \sum_{m\in M} \rho^m c_\alpha^m$. The parameters ρ_m and c_α^m, $m \in M$, are determined as follows.

- **The assignment probabilities** ρ^m, are determined through an iterative process, the detailed illustration of the process is given in EC.4. For a segment $m \in M$ we initialize the probability ρ^m as the proportion of the segment m in the overall CP pool. Then using these probabilities we solve the $\mathcal{ODP}$ defined by (4.6)-(4.8).

$$\text{Minimize} \quad \sum_{\alpha\in\mathcal{B}_\Theta} \sum_{m\in M} \rho_m \cdot c_\alpha^m \cdot s_\alpha \tag{4.6}$$

$$\text{subject to} \sum_{\alpha\in\mathcal{B}_\Theta : i\in\alpha} s_\alpha = 1 \qquad \theta \in \Theta, \quad i \in \theta \tag{4.7}$$

$$s_\alpha \in \{0,1\} \qquad \alpha \in \mathcal{B}_\Theta \tag{4.8}$$

Let $\mathbf{s}^*$ be the optimal solution of $\mathcal{ODP}$. We determine the temporary set of AUs as $\hat{B}(\Theta) = \{\alpha \in \mathcal{B}_\Theta : s_\alpha^* = 1\}$. Next, we consider a set of scenarios Ω, each having the same set of orders Θ but with random CP arrivals. We solve the matching model (4.2)-(4.5) for each of these problem instances, i.e., consider the perfect matching decisions for the given AU set. Let η_m be the number of times an AU α is served by a CP

from segment $m \in M$. We then calculate the updated probabilities as $\hat{\rho}^m = \eta_m/|\Omega|$. We continue updating the incumbent probabilities as $\rho^m = \hat{\rho}^m$ until the difference between the incumbent probabilities and the updated ratios is zero.

- **The expected assignment costs** c_α^m are calculated by sampling a set of K_m where K_m is the set of CPs in segment $m \in M$, and taking the average cost of serving the AU by these CPs, i.e., we set $c_\alpha^m = \frac{\sum_{k \in K_m} \delta_\alpha^k \cdot c_m}{|K_m|}$.

Finally, in the third step, we solve the ODP using the assignment probabilities and expected assignment costs obtained in the previous step. We then construct the final AU set with those $\alpha \in \mathcal{B}(\Theta)$ for which the assignment decision s_α is one in the optimal solution.

5 Computational Studies

In this section, we start with an overview of the data that we use in our experiments (See EC.8 for detailed data description). Also, we outline our experiment design and explore the computational and managerial insights generated from our numerical experiments.

5.1 Input Data

To assess our proposed methodology in practical scenarios we have sourced data from a leading grocery chain. The company has a well-established and sophisticated omnichannel strategy, making it an ideal setting for assessing the potential of the proposed crowdpicking model and the real-time system management policy we develop in this study. We have been provided data from a representative mid-sized store and its service areas for two weeks. The data set includes in-store sales, online orders, and the store's layout. .

- The in-store sales data includes transactions within the physical store, including details such as transaction ID, customer ID, transaction date, and list of purchased items. This data pertains to customers selected by the company from their loyalty program members, who are considered likely to participate as CPs based on their historical purchases and prior involvement in promotional campaigns.

- Compensations and time limits are defined based on the segment of the CPs, aligning with the segmentation used by the retailer (*Segment-1*, *Segment-2*, *Segment-3*, and *Segment-4*). Each segment reflects different shopping behaviors. So, compensations and time limits are tailored accordingly. For instance, *Segment-4* generally consists of customers who shop during work hours, likely to indicate that they do not have nine-to-five jobs. This segment is also characterized by frequent use of promotions and a tendency to purchase snacks, suggesting a younger demographic. Therefore, candidate CPs in this segment are assumed to be more flexible with time and willing to participate for lower compensations.

We draw on several studies to analyze the appropriate compensation amount. Research by Hornuf and Vrankar (2022) indicates that hourly wages on online platforms range from $3.78 to $20.88 per hour. Meanwhile, Le and Ukkusuri (2019) study finds that workers expect to be paid an average of $12 per hour. Although these studies differ from ours where Hornuf and Vrankar (2022) focuses exclusively on online

business and Le and Ukkusuri (2019)'s work includes additional costs such as transportation (e.g., fuel), we used these findings as a baseline for developing our compensation model and by considering these studies we set the per minute compensation between 7 cents to 12 cents. Additionally, the per-minute cost of dedicated pickers is calculated based on the minimum wage in Turkey. It is important to note that the per-minute compensation values for both CPs and dedicated pickers refer to net earnings and do not include taxes, insurance, or other supplementary costs (i.e., they are not "loaded" compensation amounts).

In addition, the time limits for customer segments are determined by using the results of the empirical study by Dayarian and Pazour (2022), which found that 30% of participants would not spend more than 15 minutes, and an additional 12% would not spend more than 30 minutes on such tasks. Dedicated pickers are added to the data model with infinite time limits and per-minute wages of the employees. Per minute compensations we use in our experiments and the time limit of each customer segment are given in Table 5.

Table 5 Segment compensations and time limit in the setting

Segment	**Per-Minute Compensation (¢)**	**Time Limit**
Segment-1	12	12
Segment-2	10	15
Segment-3	8	20
Segment-4	7	25
Dedicated Picker	6	no time limit

- Online orders data includes order placement time, picking deadline, and purchased items.
- The layout of the store is presented in EC.9, with numbers corresponding to areas containing products of different categories. The retail store is structured into 10 aisles 24 main categories and over 50 subcategories. It also includes an entrance and cash desks, which are the starting and finishing points for CPs in our model. The distances between aisles reflect the real layout of the store and travel times for CPs are calculated with this data. Similar to the study conducted by Dayarian and Pazour (2022) travel time is calculated based on the average walking speed with a cart in the store (18 m/min) , while the pick-up time per item is assumed to be 0.88 minutes for both CPs and dedicated pickers. These values are added to the total travel time when calculating CP capacities and time loads.

5.2 Experimental Design and Implementation Details

We design a comprehensive experimental setup to assess the performance of the solution approach we develop for real-time management of the proposed model and derive critical managerial insights regarding its potential. Specifically, we establish a base case setting, characterized by the problem parameters directly taken from the real-world data (see EC.8 for details). Building upon this, we develop two distinct experiment groups, each designed to examine different aspects of the system by varying a single parameter relative to the base case scenario. Experiments were implemented in Python and solved with Gurobi v10 on a 64-bit Linux workstation (Intel Xeon Gold 5418Y, 503GB RAM).

• One critical parameter that impacts the proposed system's performance is CP participation. In the first group of experiments (E1), we investigate how the system's performance is influenced by varying CP participation rates. We study problem instances where the number of these selected candidate CPs is adjusted by a factor $r \in \{0.4, 0.6, 0.8, 1.0, 1.2, 1.5\}$ relative to the number of CPs in the base case.

• In the second set of experiments (E2), we examine how variations in the allowed picking time affect picking costs. In the base case—aligned with current operational practices—each online order is assigned a fixed picking deadline. To conduct a comprehensive analysis, we vary these deadlines by adding or subtracting time increments of $\mu = -45, -30, \ldots, +45$ minutes. This experiment aims to inform decisions regarding the trade-off between picking and delivery time allocation. Reducing the time available for picking may increase flexibility for delivery scheduling and consolidation, whereas allowing more time for picking may improve order preparation quality but reduce delivery time, potentially raising delivery costs.

5.3 Analysis

5.3.1 Efficacy of the Proposed Solution Approach

We evaluate the performance of the assignment policies outlined in Section 4.2.3 and 4.2.4 and the ADP benchmark by Dayarian and Pazour (2022) (see EC.7 for detailed analysis). The effectiveness of these policies is compared in Table 6, which presents the difference in order picking costs incurred by each policy relative to the lower bound established by the clairvoyant model over a week of transactions. So, smaller values indicate a better benchmark for the model. The calculation of the **gap** is given below:

$$\text{Gap } (\%) = \frac{\text{Benchmark Cost} - \text{Clairvoyant Cost}}{\text{Benchmark Cost}} \times 100$$

The table also presents the cost performance of the three preprocessing methods—*As-Is*, *Under Sampling*, and *Relabeling*—used in the ML-based policies (BS, SO, and SU), as described in Subsection 4.2.2.

Table 6 Cost performance of the different assignment policies

Policy	**As-Is**	**Under Sampling**	**Relabeling**
Best Score (BS)	40,76%	18,46%	17,46%
Score Ordering (SO)	58,89%	15,18%	19,25%
Score And Urgency (SU)	34,45%	9,45%	**8,60%**
ADP (Dayarian and Pazour 2022)		26,36%	
Cheapest Assignment (CA)		75,37%	
Urgency Assignment (UA)		55,59%	

Results presented in Table 6 show that simple rule-based approaches, such as CA and UA, perform poorly and are inadequate for addressing the complexities of real-world problems, underscoring the need for more

sophisticated assignment policies. These heuristics rely on myopic, one-shot decisions made at the time of customer arrival, without considering how future arrivals or the available delivery time buffer. This is where the advantages of ML-based approaches (BS, SO, and SU) become evident. Based on our computational analysis, SU performs the best among these policies, followed by BS and SO. Both BS and SO rely solely on scores generated by ML models, resulting in over 30% better performance compared to simple rule-based approaches (CA and UA). However, SU demonstrates additional improvement over BS and SO, emerging as the best-performing policy in the proposed model due to the combined structure of ML scores and urgency prioritization. We also see that as an expected consequence of varying predictive power to identify attractive matchings, the performance of the ML-based methods strongly depends on the preprocessing method used. For all the ML-based policies, both the Under Sampling and Relabelling show substantial performance improvements compared to the As-Is baseline. While under-sampling and relabelling yield similar results, relabelling tends to perform better, particularly in SU policy. We attribute this advantage to relabelling being a more informed preprocessing technique. Unlike under-sampling, which randomly removes instances with zero labels, relabelling selectively modifies labels based on specific criteria such as cost and urgency, leading to more targeted and effective adjustments.

Also, the results clearly demonstrate that, although the minimum-cost model with a lookahead period outperforms simple heuristics, the ADP model still falls short when compared to ML-based approaches, particularly those involving relabeling. This indicates that, despite the inclusion of a lookahead period, the predictive and imitative capabilities of ML models significantly enhance task assignment performance, making them more effective than traditional optimization methods.

Additionally, order urgency plays a critical role in the effectiveness of the assignment policy. This is evident from both rule-based and ML-based approaches: UA outperforms CA by 20%, while SU achieves approximately 10% better performance than BS and 15% better than SO. Although the ML model provides highly accurate predictions, considering urgency is essential for minimizing reliance on dedicated pickers. While the model already incorporates 'time to delivery' as a feature, the results indicate that combining ML scores with explicit prioritization of time-sensitive orders can further improve overall system efficiency.

5.3.2 Managerial Insights

In this section, we present managerial insights derived from computational analysis. The themes were guided by the model outputs and the concerns and advantages identified in expert interviews (Section 3.2).

We start by examining the impact of participation rates on the picking cost using the problem instances from E1 experiments. Specifically, we analyze the levels of CP participation necessary to make the system economically viable. Additionally, we explore how the required number of dedicated pickers changes with the implementation of the crowdpicking system compared to the existing model. The significance of order decomposition is also evaluated, considering different basket sizes and the workload distribution among

CPs. Finally, we assess the adjustments in picking deadlines and their potential cost advantages or disadvantages to the overall order fulfillment cost by analyzing the E2 experiments.

Potential benefits of the system

Minimizing the required number of employees is crucial for reducing order fulfillment costs. However, balancing the trade-off between minimizing idle time and maintaining the desired service level remains a challenge—particularly in grocery retail, where online and offline demands have step peaks at certain times of the day. Reflecting this, several experts interviewed in Section 3.2 highlighted dynamic workforce management and the potential for labor cost reduction as key advantages of the crowdpicking model.

As shown in Figure 4, in the base case setting, the number of required store employees at different times of the day fluctuates between one and five. In contrast, the orange box represents the required number of pickers when a crowdpicking system is implemented. Notably, this system requires dedicated pickers only in the morning, with a single employee needed on 80% of the days. A significant insight is that these times coincide with lower in-store customer traffic, allowing existing staff to be reallocated for picking operations without impacting customer service. So, by leveraging the existing workforce the need for additional dedicated pickers can potentially be removed entirely.

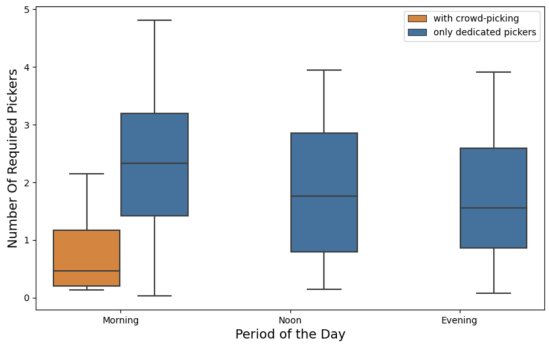


Figure 4 Required Number of Dedicated Pickers in Different Periods of the Day

The cost benefits of the system are closely tied to the availability of CPs. To provide insights into the impact of varying levels of customer engagement, we analyze the system's costs based on data generated from the E1 experiments. Figure 5 illustrates the percentage savings of the proposed model as the number of candidate CPs in the system changes. The percentage savings are calculated by comparing the expected picking costs for dedicated pickers with the costs incurred by the proposed system. The threshold (red dashed line) is determined by calculating the total time required for dedicated pickers, measured in minutes, and multiplying this by the average per-minute wage. However, in practice, it is unlikely that retailers can hire employees on such a granular per-minute basis. In a more realistic scenario, hiring would likely occur on a daily or weekly basis, at minimum, which would influence the actual cost calculation. So our analysis essentially presents a conservative estimate of the savings that model can offer.

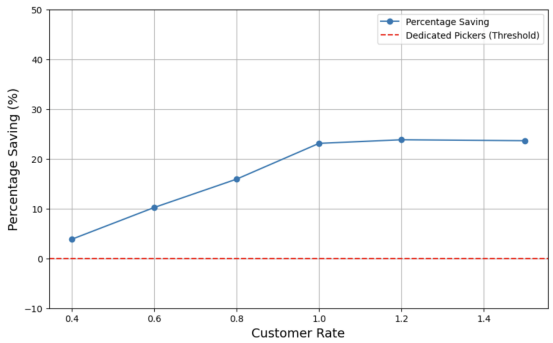


Figure 5 Change in savings based on customer participation

As we see in Figure 5, increasing the number of in-store customers beyond the base case level provided does not lead to significant improvements in cost savings at the current online order levels. Therefore, it can be inferred that, for the given online order volume, there is no pressing need to aggressively promote CP participation beyond the loyalty customers considered in the potential CP pool. However, if CP participation in the system falls short of expectations, it may result in a significant increase in the costs of the crowdpicking system. This effect is particularly pronounced when fewer than 40% of the expected CPs participate, at which point the system loses most of its cost advantages.

Considering the survey results from Dayarian and Pazour (2022) show that 65% of respondents would always or occasionally participate and parallel to this, in the feasibility study reported in Section 3.2, none of the interviewed experts considered insufficient customer participation to be a major concern. Moreover, the CPs in the base case are not randomly selected from the general customer pool, where this ratio may apply, but rather represent a targeted group identified by the company's experts based on their high likelihood of participation. Taken together, the findings suggest that the system has strong potential for achieving savings.

Impact of Order Decomposition

To enhance the efficiency of using CPs and minimize their workload, thereby increasing participation in the system, we have incorporated an order decomposition process, as elaborated in Section 4.3. Given that we utilize customers who are already present for their own shopping, we partition orders in a way that minimizes the additional time required for these customers to complete picking tasks. This approach ensures that in-store customers are not overly burdened with picking responsibilities and allows for more modular task assignments, ensuring that the picking tasks do not cause significant deviations from their preferred shopping path within the store. In particular, for orders with diverse product types, the crowdpicking system may face challenges in making cost-effective assignments due to the variety of aisles that need to be visited.

In Figure 6, we present the average picking costs for orders with varying numbers of items. The blue line represents the costs when orders are divided into multiple AUs, while the orange line reflects the costs when orders are not divided. As clearly shown in the figure, dividing orders into more manageable AUs significantly reduces picking costs, particularly for orders with more than ten items. For example, the cost savings can exceed 60% for orders containing 15 items.

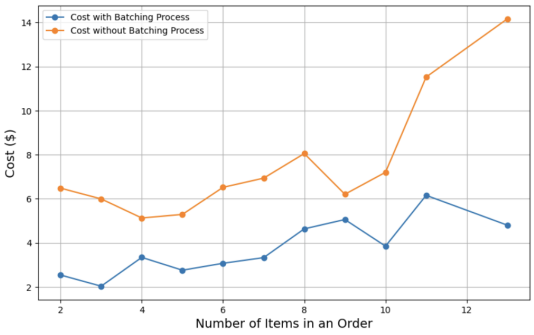


Figure 6 Cost vs. Number of Products in an Order

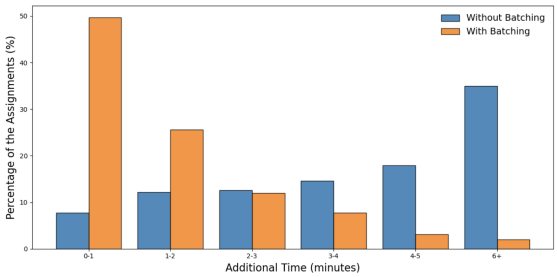


Figure 7 Average Time Load of a CP

In addition to the cost advantages, the benefits of the order decomposition process (dividing the orders) to reduce the time burden for CPs is presented in Figure 7 that illustrates the distribution of additional time loads assigned to CPs, both with and without order division. The figure shows that the majority (almost 75%) of AUs given to CPs under the order division approach result in an additional time load of up to two minutes. In contrast, when an entire order is assigned to a single CP, approximately 40% of the orders impose a time load exceeding six minutes. Overall, by dividing orders into AUs, the average additional time per CP is reduced from five minutes to three minutes. This reduction not only significantly lowers the overall picking costs but also increases the likelihood of CPs accepting the assignments.

Expert interviews (Section 3.2) also revealed concerns about increased picking time, complexity, and the risk of errors. By enabling smaller and targeted assignments, the decomposition strategy has potential to mitigate these concerns and support smoother integration of crowdpicking into real-world operations.

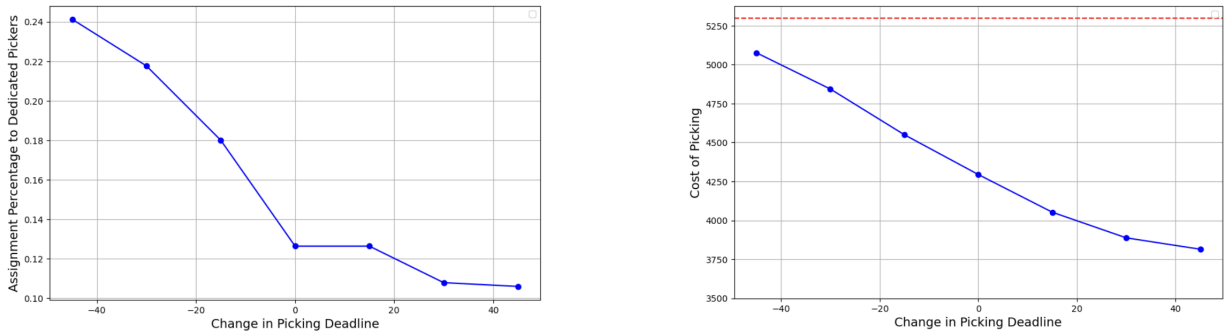
However, while decomposing customer orders into AUs enhances flexibility and responsiveness, it can also shifts complexity to the post-picking stage, particularly at the packing station. Since each AU corresponds to a single customer order, complex sorting is unnecessary; however, timely consolidation of AUs before delivery remains essential. As AUs may arrive at different times, the packing station must function as a temporary buffer. This requirement aligns with standard operations in warehouses and is unlikely to pose significant operational challenges, making it a reasonable trade-off for the gains in assignment efficiency.

Picking deadline adjustments

Two key processes drive costs in order fulfillment: picking and delivery. The total fulfillment time consists of two stages, creating a trade-off in allocating the available time between them. Longer picking deadlines offer the advantage of better utilizing in-store customers, potentially reducing the need for dedicated pickers and lowering labor costs. However, this also reduces the time window available for last-mile delivery, potentially increasing delivery costs. Conversely, shorter picking deadlines may increase picking costs but allow more time for delivery, possibly enhancing efficiency in that phase. In Experiment E2, we focus on understanding how changes in picking times impact picking costs and the reliance on dedicated pickers.

Figure 8 illustrates how the rate of picking assignments to CPs and the associated picking costs vary with changes in the adjustments in the picking deadline. These adjustments involve either increasing or

decreasing the available picking time at the expense of the available delivery time, relative to the base case scenario. Positive values indicate how much the time allocated for picking is increased and the negative values indicate how much it is shortened. For the picking costs, the red line in Figure 8b shows the costs with no crowdshipping as a reference. Firstly, we observe that the current time allocation between the picking and the delivery operations (marked with 0 in the x axis), operates successfully by requiring dedicated pickers for only 10% of the AUs. As an interesting insight, the Figure shows that there are no significant additional benefits of extending picking deadlines to reduce reliance on dedicated pickers. However, we also see that when deadlines are shortened, there is an observable shift in dynamics: the system witnesses a rapid increase in the percentage of tasks allocated to dedicated pickers. Shorter deadlines require an increased reliance on dedicated pickers. This rapid increase in dedicated pickers and potentially expensive assignments CPs to meet tighter deadlines drives up picking costs, nearly to the level where the cost advantage of the system diminishes. This highlights that any adjustments to the time allocated for picking operations, particularly reducing it, could result in costly outcomes. Therefore, changes in process times should be approached cautiously, ensuring a careful balance between the trade-offs across different operational phases.



(a) Assignment percentage to dedicated pickers

(b) Picking costs

Figure 8 Impact of deadlines on system efficiency

6 Discussion and Future Work

In this study, we introduced crowdpicking, a novel in-store fulfillment system that dynamically assigns online order picking tasks to in-store customers. Although the concept has not yet been implemented in practice, expert interviews with senior professionals from the omnichannel retail sector (see Section 3.2) revealed strong interest in its potential, along with several perceived advantages and barriers.

Building on these insights, we formulated a real-time task assignment problem and proposed a machine learning–based decision model within a sequential decision-making framework. The model leverages optimized assignments to make fast and adaptive decisions under uncertainty. Experimental results show that our approach yields significant performance improvements over state-of-the-art benchmark algorithms.

Beyond task allocation, we also proposed an order-splitting algorithm that divides orders into smaller suborders to increase the likelihood of customer participation. Integrating this mechanism into our overall assignment framework resulted in additional, substantial gains in fulfillment efficiency. To ground our findings in practice, we conducted a series of experiments using real operational data from a leading omnichannel grocery retailer, including online and in-store transactions, and detailed store layout. These experiments revealed the potential cost and efficiency gains of crowdpicking under various real-world scenarios.

While this study offers valuable insights and contributes significantly to the field of ship-from-store fulfillment, several limitations and opportunities for further research remain. Firstly, although the theoretical models provide strong guidance on customer participation and optimal compensation strategies, empirical validation through pilot studies or structured surveys is essential. Such validation can help capture real customer behavior, assess responsiveness to various incentives, and refine model assumptions accordingly.

Secondly, the assumption that CPs always accept assigned tasks simplifies the problem but may limit applicability. In practice, task rejections may occur due to time constraints, insufficient compensation, or CP availability. Future models should incorporate probabilistic acceptance behaviors and explore adaptive assignment strategies that can respond to rejections. Real-world experiments will be critical for designing and testing such rejection-handling mechanisms, ensuring robustness under practical uncertainties.

Parallel to this, future studies could explore dynamic fallback mechanisms that reassign rejected tasks in real time, potentially balancing customer preferences with system efficiency. Furthermore, integrating predictive models for acceptance likelihood could enhance proactive assignment decisions in operations.

While this study focuses on cost minimization, assignments could also promote sales by guiding CPs toward products they may add to their own shopping. Developing system management policies that account for these opportunity sales could significantly enhance the benefits of the proposed order picking model, not only by reducing costs but also by increasing revenues.

Also, an important perspective for future work lies in exploring the downstream fulfillment implications of order decomposition. In our current approach, each AU is tied to a single customer order, allowing all picked items to be easily consolidated at the packing station without requiring sorting. This process can be managed with basic labeling, enabling smooth integration into existing in-store operations. However, future research could investigate the potential benefits of bundling AUs from multiple customer orders into shared picking tasks. While such inter-order bundling may further reduce picking costs by improving route efficiency—especially in dense store layouts or during peak periods—it would also require a more complex operational design at the packing stage, including additional tracking and sorting capabilities. This trade-off between picker efficiency and packing complexity presents a promising direction for future studies involving joint optimization of picking and packing processes.

Finally, in this study, we present percentage-based cost savings by referencing country-specific minimum wage and existing CS compensation schemes. To build a more realistic and actionable cost assessment,

future work should incorporate granular cost components such as the number and salaries of dedicated pickers per period and empirically derived customer compensation expectations. Importantly, maintaining high service levels in online order fulfillment depends on a stable and responsive workforce, yet labor contracts typically span longer periods, making the assumption of unlimited picker availability unrealistic. Overstaffing inflates labor costs, while understaffing risks service level. Therefore, a risk-benefit analysis to determine the optimal number of dedicated pickers is crucial for aligning cost with operational reliability.

Acknowledgments

This study has been supported by the European Research Council (ERC) under grant number 101076231 as a part of the GoodMobility project (https://goodmobility.ku.edu.tr).

References

A. M. Arslan, N. Agatz, L. Kroon, and R. Zuidwijk. Crowdsourced delivery—a dynamic pickup and delivery problem with ad hoc drivers. *Transportation Science*, 53(1):222–235, 2019.

K. Asdemir, V. S. Jacob, and R. Krishnan. Dynamic pricing of multiple home delivery options. *European Journal of Operational Research*, 196(1):246–257, 2009.

O. Baron, A. A. Cire, and S. K. Savaser. Decentralized online order fulfillment in omni-channel retailers. *Production and Operations Management*, 33(8):1719–1738, 2024.

A. Bayram and B. Cesaret. Order fulfillment policies for ship-from-store implementation in omni-channel retailing. *European Journal of Operational Research*, 294(3):987–1002, 2021.

D. R. Bell, D. Corsten, and G. Knox. Unplanned category purchase incidence: Who does it, how often, and why. 2013.

J. Bhowmick, J. Pazour, and I. Dayarian. Store fulfillment with autonomous mobile robots and in-store customers. *IISE Transactions*, pages 1–18, 2025.

T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

J.-F. Cordeau and G. Laporte. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research Part B: Methodological*, 37(6):579–594, 2003.

I. Dayarian and J. Pazour. Crowdsourced order-fulfillment policies using in-store customers. *Production and Operations Management*, 31(11):4075–4094, 2022.

C. Dethlefs, M. Ostermeier, and A. Hübner. Rapid fulfillment of online orders in omnichannel grocery retailing. *EURO Journal on Transportation and Logistics*, 11:100082, 2022.

R. M. Difrancesco, I. M. van Schilt, and M. Winkenbach. Optimal in-store fulfillment policies for online orders in an omni-channel retail environment. *European Journal of Operational Research*, 293(3):1058–1076, 2021.

S. Gallino and A. Moreno. *Operations in an omnichannel world: Introduction*. Springer, 2019.

J. Goedhart, R. Haijema, R. Akkerman, and S. de Leeuw. Replenishment and fulfilment decisions for stores in an omni-channel retail network. *European Journal of Operational Research*, 311(3):1009–1022, 2023.

H. Haddon and J. Jargon. Consumers love food delivery. restaurants and grocers hate it. *The Wall Street Journal*, 2019.

L. Hornuf and D. Vrankar. Hourly wages in crowdworking: A meta-analysis. *Business & Information Systems Engineering*, 64(5):553–573, 2022.

Y. Hou, B. Li, L. Li, and J. Liu. A density-based under-sampling algorithm for imbalance classification. In *Journal of physics: Conference series*, volume 1302, page 022064. IOP Publishing, 2019.

A. Hübner, J. Hense, and C. Dethlefs. The revival of retail stores via omnichannel operations: A literature review and research framework. *European Journal of Operational Research*, 302(3):799–818, 2022.

A. H. Hübner, H. Kuhn, and J. Wollenburg. Last mile fulfilment and distribution in omni-channel grocery retailing: a strategic planning framework. *International Journal of Retail & Distribution Management*, 44(3), 2016.

S. Jiu. Robust omnichannel retail operations with the implementation of ship-from-store. *Transportation Research Part E: Logistics and Transportation Review*, 157:102550, 2022.

A. Kobren, C. H. Tan, P. Ipeirotis, and E. Gabrilovich. Getting more for less: Optimized crowdsourcing with dynamic tasks and goals. In *Proceedings of the 24th international conference on world wide web*, pages 592–602, 2015.

T. V. Le and S. V. Ukkusuri. Modeling the willingness to work as crowd-shippers and travel time tolerance in emerging logistics services. *Travel Behaviour and Society*, 15:123–132, 2019.

S. M. Lundberg, G. Erion, H. Chen, A. DeGrave, J. M. Prutkin, B. Nair, R. Katz, J. Himmelfarb, N. Bansal, and S.-I. Lee. Explainable ai for trees: From local explanations to global understanding. *Cornell University Library*, 2019.

W. Y. Mak. Peer-to-peer crowdshipping as an omnichannel retail strategy. *Transportation Research Part E: Logistics and Transportation Review*, 116:40–54, 2018.

R. Mallah. A comprehensive review and comparative study data imbalance in machine learning. *International Research Journal of Modernization in Engineering Technology and Science*, 2023.

S. Milligan, I. Davies, B. Yalabik, M. Celik, and B. Squire. Evolutions of omni-channel fulfillment performance: An in-depth case study in grocery retailing. *Journal of Operations Management*, 2025.

S. Mou. In-store order fulfilment in omni-channel supermarkets with heterogeneous workforce: A bi-objective optimisation approach. *Computers & Industrial Engineering*, 171:108516, 2022.

F. Neves-Moreira and P. Amorim. Learning efficient in-store picking strategies to reduce customer encounters in omnichannel retail. *International Journal of Production Economics*, 267:109074, 2024.

S. R. Pandala. Shankarpandala/lazypredict. Available at https://github.com/shankarpandala/lazypredict, 2024. Last Accessed: Feb 04, 2024.

N. O. Pietri, X. Chou, D. Loske, M. Klumpp, and R. Montemanni. The buy-online-pick-up-in-store retailing model: Optimization strategies for in-store picking and packing. *Algorithms*, 14(12):350, 2021.

W. B. Powell. *Approximate Dynamic Programming: Solving the curses of dimensionality*, volume 703. John Wiley & Sons, 2007.

W. B. Powell. A unified framework for stochastic optimization. *European Journal of Operational Research*, 275(3): 795–821, 2019.

D. Rigby. The future of shopping. *Harvard business review*, 89(12):65–76, 2011.

S. Ropke and D. Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40(4):455–472, 2006.

A. Seghezzi, R. Mangiaracina, and A. Tumino. E-grocery logistics: exploring the gap between research and practice. *The International Journal of Logistics Management*, 2022a.

A. Seghezzi, C. Siragusa, and R. Mangiaracina. Enhancing in-store picking for e-grocery: an empirical-based model. *International Journal of Physical Distribution & Logistics Management*, 52(4):301–323, 2022b.

Statista. Online Grocery Shopping in the United States. Available at https://www.statista.com/topics/1915/online-grocery-shopping-in-the-united-states/#topicOverview, 2024. Last Accessed: Sep 08, 2024.

J. Tanha, Y. Abdi, N. Samadi, N. Razzaghi, and M. Asadpour. Boosting methods for multi-class imbalanced data classification: an experimental review. *Journal of Big Data*, 7:1–47, 2020.

The Economist. Has e-commerce peaked?, 2023. URL https://www.economist.com/business/2023/08/04/has-e-commerce-peaked. Accessed: August 28, 2024.

M. Vazquez-Noguerol, J. A. Comesaña-Benavides, S. Riveiro-Sanroman, and J. C. Prado-Prado. A mixed integer linear programming model to support e-fulfillment strategies in warehouse-based supermarket chains. *Central European Journal of Operations Research*, 30(4):1369–1402, 2022.

C. Wildt, F. Weidinger, and N. Boysen. Picker routing in scattered storage warehouses: an evaluation of solution methods based on tsp transformations. *OR Spectrum*, pages 1–32, 2024.

B. Yıldız. Express package routing problem with occasional couriers. *Transportation Research Part C: Emerging Technologies*, 123:102994, 2021.

L. Y. Zatz, A. J. Moran, R. L. Franckle, J. P. Block, T. Hou, D. Blue, J. C. Greene, S. Gortmaker, S. N. Bleich, M. Polacsek, A. N. Thorndike, and E. B. Rimm. Comparing online and in-store grocery purchases. *Journal of Nutrition Education and Behavior*, 53(6):471–479, 2021.

E-Companion

EC.1 Extension of Planned Shopping Lists

In our analysis, we initially constructed customers' shopping lists based on their planned purchases. However, in realistic scenarios, customers may expand their decisions once they are physically present in the store. To capture this behavior, we investigated the impact of such deviations—where customers shop from additional aisles beyond their initial plans—on the total time spent in the system and their ability to complete assigned crowdpicking tasks on time.

Figure EC.1 illustrates the effect of visiting additional aisles on both the total time spent in the system (blue line) and the delay rate of order assignments (orange line). As shown by the blue line, the average time spent in the system increases in an approximately linear fashion with the number of unplanned aisles visited. This outcome is intuitive: the more aisles a customer visits, the longer they remain in the system. However, a small but meaningful deviation from this trend is observed. In certain cases, adding additional items can slightly reduce the additional time spent for crowdpicking. This occurs when the newly added items are located along the customer's existing route, thus improving overall route efficiency. This explains why the increase in time does not follow a perfectly linear pattern.

While adding a few extra items may seem operationally minor, it can still pose a risk to the timely preparation of orders. Our assignment model addresses this risk by incorporating the estimated route completion time—based on both the original shopping list and the assigned picking tasks—into its feasibility checks. Nevertheless, when shopping lists are extended, service level violations may occur.

To analyze the impact of unexpected route extensions, we simulated the effect of randomly adding aisles to customer paths and measured the resulting order delays. As shown by the orange line, the delay rate remains minimal for small deviations: only 0.1% of orders are delayed when one additional aisle is added, and 0.3% when two aisles are added.

This observation aligns with prior research, which suggests that such minor deviations are typical. Specifically, Bell et al. (2013) report that approximately 20% of purchased categories are unplanned, indicating that customers frequently stray slightly from their original shopping lists. Considering that the store layout consists of 10 aisles and 24 product categories, adding 1–2 unplanned aisles appears to be a realistic and representative behavior observed in practice. The results indicate that as long as such deviations remain within this limited range, the overall service level is unlikely to be significantly affected.

However, the risk of delay rises significantly beyond this threshold. When customers visit three or more unplanned aisles, the delay rate increases sharply, averaging around 10% and reaching up to 15% in extreme cases. Although visiting four or five unplanned aisles—equivalent to covering nearly half the store outside the planned route—is rare, it can create problems for the timely fulfillment of customer demands when it

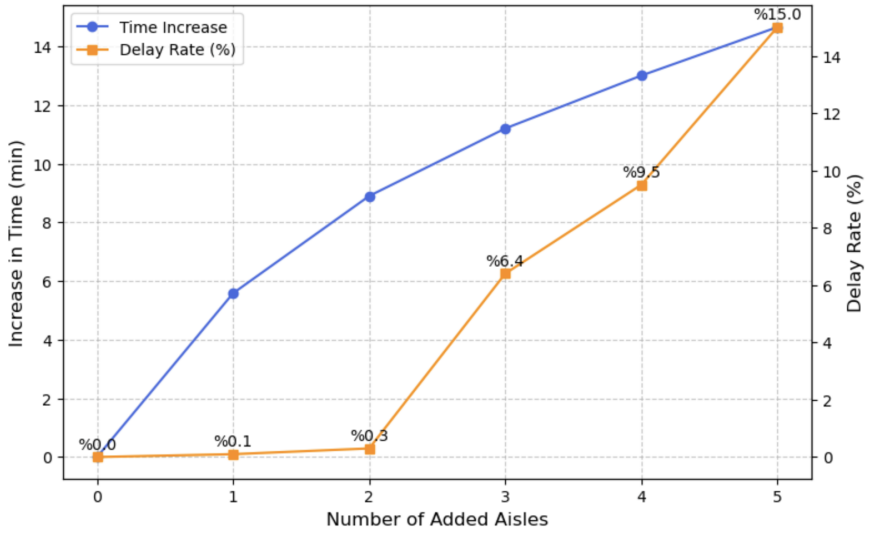


Figure EC.1 Increase in Time Spent by Number of Added Aisles

occurs. While such cases are uncommon, stores seeking to be fully resilient to these outliers may choose to incorporate a small time buffer. Our results show that a 15-minute buffer is sufficient to eliminate picking delays even under these extreme deviations, offering a safeguard without compromising efficiency in typical scenarios.

EC.2 Model Performance Results

Table EC.1 Comparison of Different Classification Algorithms in As-Is Training Data

ML Model Performance				
Model	Balanced Accuracy (ROC-AUC)	F1 Score	Precision	Recall
Perceptron	0,8534	0,9544	0,5896	0,8734
PassiveAggressiveClassifier	0,8515	0,9579	0,6042	0,8715
GaussianNB	0,8309	0,7892	0,5141	0,8309
NearestCentroid	0,7895	0,7729	0,5120	0,7895
LGBMClassifier	0,6696	0,9875	0,6805	0,6696
ExtraTreeClassifier	0,6578	0,9851	0,6262	0,6578
QuadraticDiscriminantAnalysis	0,6451	0,5521	0,5061	0,6501
BaggingClassifier	0,6451	0,9880	0,7021	0,6451
RandomForestClassifier	0,6407	0,9889	0,7510	0,6407
DecisionTreeClassifier	0,6405	0,9853	0,6254	0,6405
XGBClassifier	0,6222	0,9873	0,6805	0,6222
AdaBoostClassifier	0,6157	0,9886	0,7636	0,6157
ExtraTreesClassifier	0,5931	0,9882	0,7763	0,5931
LogisticRegression	0,5883	0,9884	0,8261	0,5883
BernoulliNB	0,5859	0,9715	0,5261	0,5859
LinearDiscriminantAnalysis	0,5852	0,9876	0,7256	0,5852
CalibratedClassifierCV	0,5708	0,9881	0,8642	0,5708
KNeighborsClassifier	0,5704	0,9876	0,7703	0,5704
LinearSVC	0,5658	0,9880	0,9019	0,5658
LabelSpreading	0,5612	0,9854	0,6083	0,5612
LabelPropagation	0,5612	0,9854	0,6083	0,5612
DummyClassifier	0,5	0,9853	0,4951	0,5
SVC	0,4999	0,9852	0,4951	0,4999

Table EC.2 Comparison of the Different Classification Algorithms with Under-Sampling

ML Model Performance				
Model	Balanced Accuracy (ROC-AUC)	F1 Score	Precision	Recall
LinearSVC	0,9546	0,9620	0,9286	0,9746
CalibratedClassifierCV	0,9546	0,9620	0,9286	0,9746
RandomForestClassifier	0,9392	0,9680	0,9444	0,9692
LogisticRegression	0,9361	0,9497	0,9091	0,9661
SVC	0,9361	0,9661	0,9091	0,9661
LinearDiscriminantAnalysis	0,9361	0,9497	0,9091	0,9661
SGDClassifier	0,9319	0,9436	0,9000	0,9619
GaussianNB	0,9319	0,9436	0,9000	0,9619
QuadraticDiscriminantAnalysis	0,9319	0,9436	0,9000	0,9619
PassiveAggressiveClassifier	0,9307	0,9555	0,9224	0,9607
RidgeClassifier	0,9522	0,9433	0,9025	0,9522
RidgeClassifierCV	0,9226	0,9428	0,9058	0,9426
XGBClassifier	0,9214	0,9548	0,9331	0,9414
LGBMClassifier	0,9214	0,9548	0,9331	0,9414
BaggingClassifier	0,9214	0,9548	0,9331	0,9414
Perceptron	0,9183	0,9367	0,8958	0,9383
AdaBoostClassifier	0,9129	0,9424	0,9100	0,9329
ExtraTreesClassifier	0,9117	0,9543	0,9403	0,9317
DecisionTreeClassifier	0,9117	0,9543	0,9403	0,9317
ExtraTreeClassifier	0,9033	0,9418	0,9153	0,9233
KNeighborsClassifier	0,8967	0,9167	0,8757	0,8967
LabelSpreading	0,8635	0,9082	0,8784	0,8635
LabelPropagation	0,8635	0,9082	0,8784	0,8635
NearestCentroid	0,8091	0,8688	0,8218	0,8091
BernoulliNB	0,7921	0,8456	0,7815	0,7921
NuSVC	0,7801	0,8641	0,8373	0,7801
DummyClassifier	0,5000	0,6648	0,3831	0,5000

Table EC.3 Comparison of Different Classification Algorithms with Relabeling preprocessing

Model	Balanced Accuracy (ROC-AUC)	F1 Score	Precision	Recall
XGBClassifier	0,9939	0,9937	0,9936	0,9939
LGBMClassifier	0,9926	0,9924	0,9922	0,9926
RandomForestClassifier	0,9894	0,9892	0,9891	0,9894
BaggingClassifier	0,9858	0,9856	0,9854	0,9858
ExtraTreesClassifier	0,9850	0,9847	0,9845	0,9850
DecisionTreeClassifier	0,9829	0,9829	0,9829	0,9829
LabelPropagation	0,9674	0,9672	0,9670	0,9674
LabelSpreading	0,9665	0,9663	0,9661	0,9665
ExtraTreeClassifier	0,9529	0,9528	0,9526	0,9529
KNeighborsClassifier	0,9520	0,9519	0,9517	0,9520
SVC	0,9388	0,9384	0,9382	0,9388
AdaBoostClassifier	0,8855	0,8864	0,8883	0,8855
LinearSVC	0,8724	0,8734	0,8752	0,8724
CalibratedClassifierCV	0,8718	0,8725	0,8736	0,8718
LogisticRegression	0,8713	0,8721	0,8733	0,8713
NuSVC	0,8679	0,8688	0,8708	0,8679
SGDClassifier	0,8668	0,8676	0,8687	0,8668
LinearDiscriminantAnalysis	0,8578	0,8593	0,8663	0,8578
RidgeClassifier	0,8578	0,8593	0,8663	0,8578
RidgeClassifierCV	0,8574	0,8589	0,8656	0,8574
NearestCentroid	0,8483	0,8498	0,8567	0,8483

EC.3 ML Model Feature Explanations

Table EC.4 ML model features

ID	Feature Name	Explanation
1	Segment Code	Segment of the CP
2	Time Period	The period of the day when the assignment is performed, categorized as morning, noon, or evening
3	Day	The day of the week when the assignment occurs
4	Order Placement Period	Online order arrival period categorized as morning, noon, or evening
5	Deadline Period	The deadline period to collect the order categorized as morning, noon, or evening
6	Time in the System	Time elapsed since the arrival of the order
7	Time to Deadline	Time difference (in minutes) between the current time and the picking deadline of the order
8	Deadline Urgency	The urgency category of the AUs (order deadline) remaining time to pickup deadline, classified as urgent, moderate, or extended
9	Picking Lead Time	Time to picking deadline when the order arrived
10	Time Load	Extra time (in minutes) needed for collecting the AU
11	Cost	Cost of giving the AU to the CP
12	Difference to Cheapest	Cost difference between the cheapest available option among all CP-AU pairs
13	Cheapest	Cheapest available price option among CP-AU pairs
14	Remaining Orders	Number of orders that have arrived but have not yet been collected
15	Remaining Earlier Orders	Number of orders that have arrived but not collected, with an earlier deadline
16	Orders up to now	Total number of online orders up to current time
17	Customers up to now	Total number of in-store customers up to current time

EC.4 Iterative Propobability Update Process

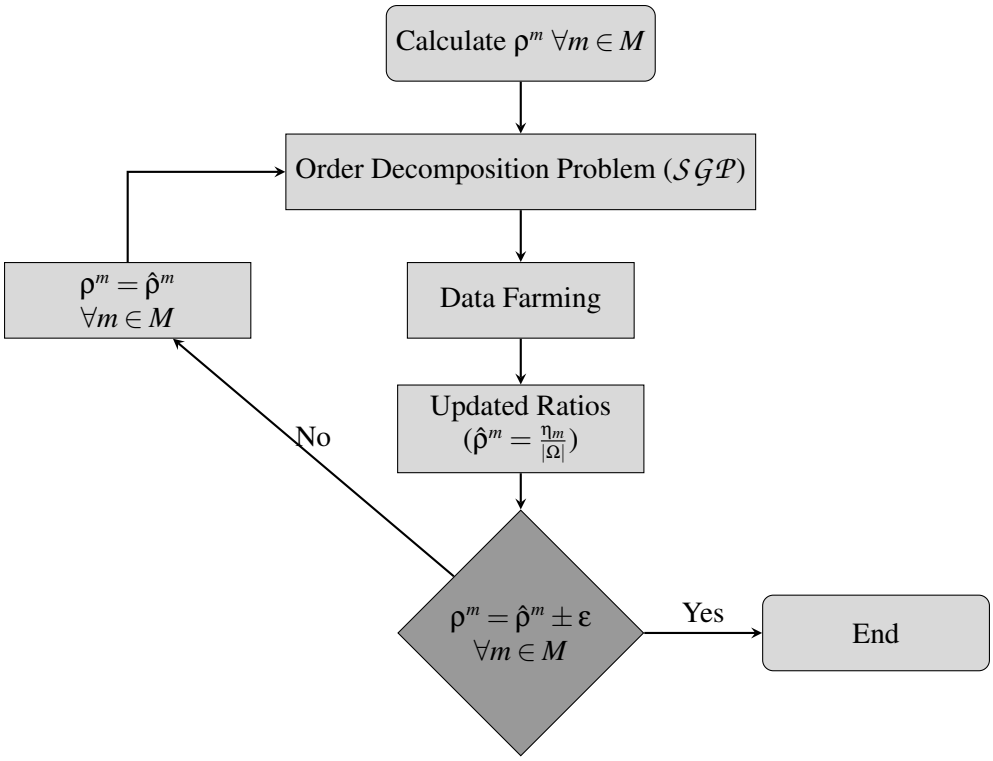


Figure EC.2 Iterative Probability Update Process

EC.5 Simulation Model

We have developed a simulation algorithm for real-time assignment, which we evaluate the cost of various policies. This algorithm operates in two stages: the first stage involves selecting the threshold that minimizes cost within the training dataset, and the second stage evaluates different strategies with the selected threshold. The parameters and sets for the assignment are explained in detail below for a predefined threshold in Algorithm 1.

The algorithm operates with 4 inputs. These inputs are (1) Θ and (2) K (3) T correspond to the online orders to serve, set CPs and operating hours of the stores respectively as described in 4.1. And (4) Υ is the selected threshold for the ML score. Note that this threshold is used to determine the reliance to the ML model; i.e., if the threshold is 0.9, the model filters the CP-AU pairs lower than 0.9 or higher. The overall output of the algorithm is to find the optimal $y_{\alpha k}$ for each decision epoch where $y_{\alpha k}$ is the binary decision variable indicates whether AU α assigned to CP k $\in$ K.

Steps (1), (2), and (21) define the progress of the system within the algorithm. Specifically, the parameter t represents the time of day. The algorithm progresses by incrementing t by one minute in each iteration at Step (21), starting from the beginning of the operating hour as established in Step (1), and continues during the store operating hours as defined in Step (2). The state of the algorithm is updated based on the available CPs for Θ_t within Θ. Specifically, Step (3) checks whether the state of the algorithm has changed.

- **If the state is changed**, for each open order at time t, we check the deadline of assignments in Step (5). Although a specific $\mathcal{BP}$ solution is proposed in Section 4.3, the algorithm can be altered based on different operating rules and priorities. To simplify the algorithm, we collect AU configurations from the proposed algorithm in Section 4.3 as $\hat{B}(\Theta)$. We then proceed with the assignment decisions outlined in Steps 9-12. We elaborate the decision making path for the *Score and Urgency*. The actions taken in these steps vary based on the selected assignment strategy. We can design all six decision policies by updating Steps (9) through (12) in Algorithm 1.

  — **Best Score**: Update Step (11) as *'Sort AUs by $P_{\alpha k}$'*.

  — **Cheapest Assignment**: Set $P_{\alpha k} = 1$ in Step (9) and update Step (11) as *'Sort AUs by $c_{\alpha k}$'* where $c_{\alpha k}$ is the cost of giving AU $\alpha \in \hat{B}(\Theta)$ to the CP $k \in K$.

  — **Score Ordering**: Update $P_{\alpha k}$ such that $P_{\alpha k} \sim \text{Uniform}(0,1)$, to denote that P_α is chosen randomly from a uniform distribution between 0 and 1 in Step (9).

  — **Urgency Assignment**: Set $P_{\alpha k} = 1$ in Step (9).

- **If there is no change in the state**, the algorithm proceeds to Step (13), where it checks for any orders that are delayed based on their deadlines. If a delay is detected (10 minutes to deadline), the system automatically give the AUs to the dedicated pickers, along with the corresponding cost. Following this, the time is updated in Step (17).

Algorithm 1: Assignment Strategy

Input: Θ, K, Υ, T
Output: $C(S,\alpha)$

$t \leftarrow \min(T)$
while $t \in T$ **do**
  if $K_{\theta_t} \neq K_{\theta_{t-1}}$ **then**
    for $\theta \in \Theta_t$ **do**
      if $\delta_\alpha^k > 10$ **then**
        Calculate $\hat{B}(\Theta)$
        for $\alpha \in \hat{B}(\Theta)$ **do**
          for $k \in K_\theta$ **do**
            Compute ML score for $(\alpha,k) \leftarrow P_{\alpha k}$
            Filter AUs with $P_{\alpha k} \leq i$
            Sort AUs by $(t - d_\alpha)$
            if $\delta_\alpha^k \leq \bar{\tau}_k$ **then**
              Set $y_{\alpha k} = 1$
              $k \rightarrow$ CP k
              $C(S,\alpha) + \delta_\alpha^k * c_k$
    for $\alpha \in \hat{B}(\Theta)$ **do**
      if $(t - d_\alpha) \leq 10$ **then**
        Set $y_{\alpha k} = 1$
        $k \rightarrow$ Dedicated Picker
        $C(S,\alpha) + \bar{c}_\alpha$
  $t \leftarrow t + 1$ minute
return $C(S,\alpha)$

EC.6 Robustness Analysis under Uncertain Cost Information

In our main model, we assumed that customer compensation costs are precisely known. While this assumption simplifies the problem and is standard in many studies, it does not fully reflect real-world uncertainty. To strengthen the validity of our approach, we also evaluated model performance under a setting where cost values are not known exactly.

In these experiments, we model the uncertainty in assignment costs as random variables with known means and standard deviations ranging from 5% to 50% of the mean, representing different levels of uncertainty in expected costs.

We assume a setting in which the task to be offered to a customer is initially selected based on the expected assignment costs. Once the offer is made, the customer responds by stating the compensation they would require to complete the task. Upon learning this actual cost, the grocer then reevaluates the assignment decision, either accepting the customer's compensation demand or revoking the assignment. Since our proposed model can instantaneously update the assignment decision, it allows us to directly assess whether the model's initial decision to assign the AU to that customer remains valid given their declared cost expectation.

We also incorporated the same cost uncertainty structure into the benchmark model proposed by Dayarian and Pazour (2022). However, since their model operates with a lookahead mechanism, it may misassign customers or overlook better assignment opportunities if it only re-evaluates suitability upon customer arrival. To ensure a fair comparison of solution quality, we adapted the benchmark by assuming that customer cost expectations are revealed at the beginning of the lookahead window and remain fixed throughout. In practice, this creates a limitation: unlike our ML-based policy, which can work in real-time, the ADP cannot revise its decisions within the lookahead period if customer expectations are only revealed upon arrival. Nonetheless, we proceed with this approximation to generate a strong benchmark to assess the solution qualities of our model.

As shown in the figure, the proposed model consistently outperforms the benchmark model across all deviation levels. While the optimality gap for the proposed model remains relatively low—ranging from 6.7% to 10.1%—the benchmark model shows significantly larger gaps, varying between 15.3% and 27.1%. This indicates that the proposed model yields solutions closer to the lower bound, demonstrating better robustness and solution quality under cost uncertainty.

On the other hand, although the proposed model performs better overall, some fluctuations in its gap—particularly at mid-to-high deviation levels (%20-%50)—can be attributed to the inherent nature of the ML component. Specifically, discrepancies between training data and real-time realizations of customer behavior may lead to suboptimal decisions. This does not undermine the model's validity but highlights the importance of retraining or tuning the learning component if the model is to be adopted in dynamic, real-world environments. Notably, the observed deviations do not drastically degrade performance, suggesting the model's robustness in the face of moderate cost uncertainty.

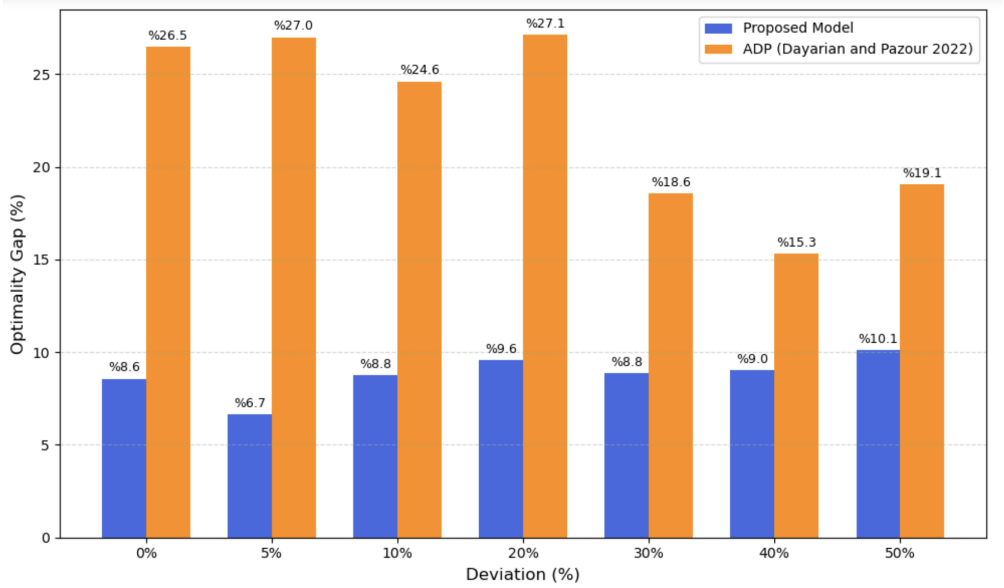


Figure EC.3 Cost Performance under Varying Levels of Uncertainty

EC.7 Sensitivity Analysis of Machine Learning Policy versus Approximate Dynamic Programming

To complement the performance comparison between the ML-based policy and ADP, this section presents a sensitivity analysis that explores the conditions under which the proposed ML approach consistently outperforms the ADP benchmark. We vary key input parameters and data characteristics—such as deadline duration, in-store customer availability, to examine the robustness and adaptability of both approaches.

To start with, we compared the average values of additional time, cost, and assignment duration (time after arrival) across the two approaches. As expected, the proposed model results in lower average costs and slightly lower additional times compared to the ADP benchmark, as shown in Figure EC.4. However, when examining the relative improvements, the most significant reduction is observed in cost (-38%), followed by additional time (-11%), while assignment time shows an increase of +27.3% under the proposed model.

This indicates that the proposed ML-based approach not only achieves more cost-efficient assignments but also aligns better with the planned routes, resulting in fewer deviations. The key advantage emerges from the model's ability to incorporate customer-specific characteristics and anticipate future arrivals during task assignment. In contrast, the ADP benchmark performs quicker assignments by selecting the best available match from the current shopper pool, with limited consideration of potential future opportunities. As illustrated in Figure EC.4, ADP leads to shorter assignment times (~45 hour) compared to the proposed model (~1 hour 10 minutes), but this speed comes at the cost of more expensive assignments.

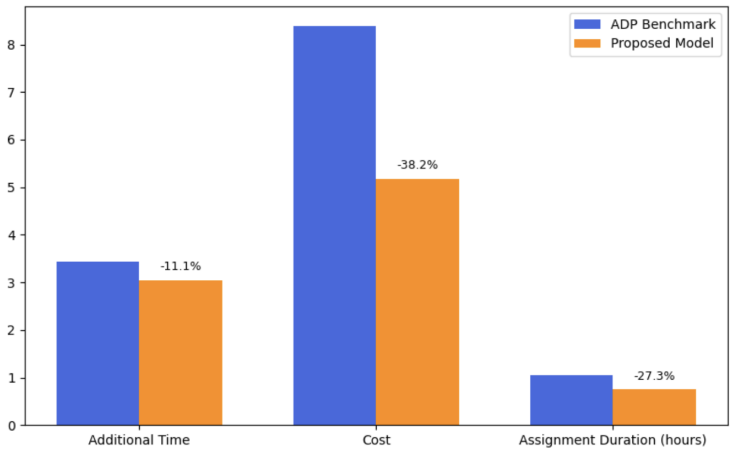


Figure EC.4 Comparison of Different Assignment Policies

Additionally, Figure EC.5 shows the total cost incurred under the ADP benchmark and proposed models, categorized by the available deadline window for each order. The x-axis groups orders based on their urgency level, ranging from *Urgent* (less than 1 hour until delivery) to *Long* (more than 3 hours). The y-axis indicates the total assignment cost within each group.

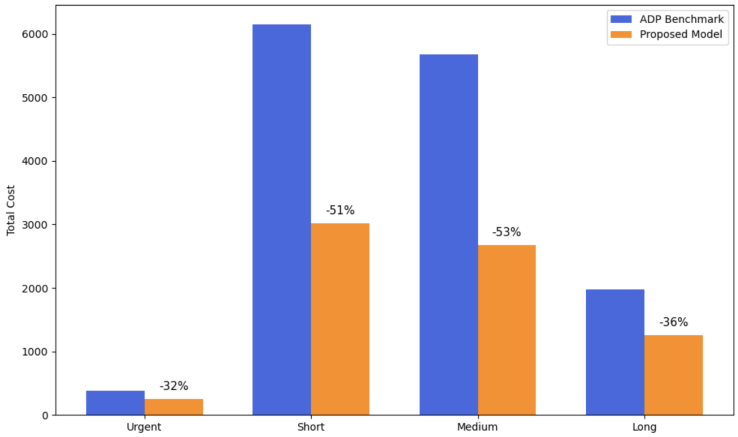


Figure EC.5 Comparison of Costs by Deadline Group

As the figure illustrates, the performance gap between the models varies across deadline intervals. The most significant cost difference is observed in the *Medium* group (2–3 hours), where the proposed ML-based model achieves a cost reduction of over 50%. This group also contains the largest number of orders, which makes this improvement particularly impactful.

The *Short* group (1–2 hours) also benefits considerably from the ML model, though the improvement is slightly lower compared to the Medium group. Surprisingly, the *Long* group exhibits a smaller relative gain despite the longer assignment window. A closer inspection reveals that many orders in this group are placed during late-night hours when stores are closed, effectively limiting the true assignment flexibility. As a result, both models struggle to find available CPs, which narrows the performance gap.

Finally, the *Urgent* group, comprising orders with less than 1 hour to be fulfilled, shows the smallest difference between models. This is expected since both models are heavily constrained and must make near-immediate assignments, reducing the ML model's opportunity to optimize.

EC.8 Data Description

Due to confidentiality constraints, we are unable to share the full dataset used in our study. However, we provide aggregated analyses that preserve data privacy while offering key statistics to describe it and support future research.

The data consists of approximately 800 online orders and 6,000 in-store visits. Each in-store visit is counted as a unique instance, corresponding to approximately 1,800 unique customers. Since the study focuses exclusively on loyalty program members, their visit frequency over a two-week period could be closely monitored. On average, a loyalty program member visits the store 3.3 times within this period. Figure EC.6 illustrates the distribution of customer visit frequencies. As shown in the Figure, nearly 90% of customers visit the store fewer than five times. However, there are notable outliers, with some customers visiting almost daily.

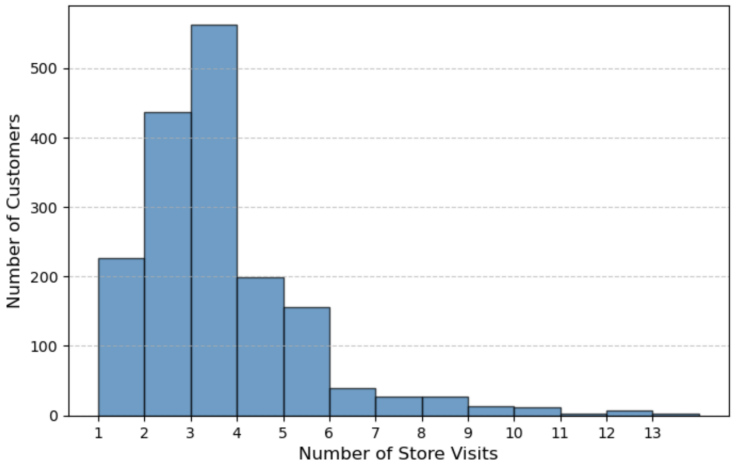
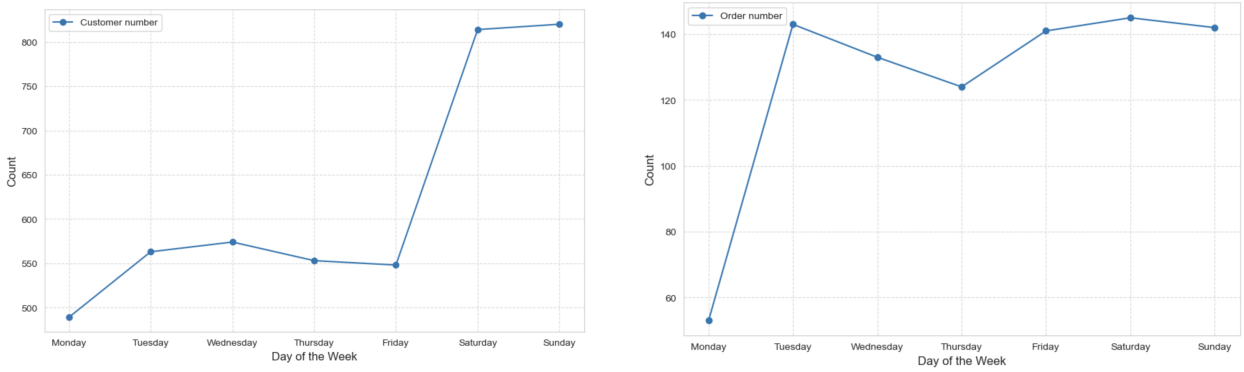


Figure EC.6 Distribution of Customer Visit Frequencies

The distribution of order numbers and in-store customers across different days of the week is a crucial factor in the analysis. Figure EC.7 illustrates the distribution of in-store customers and online orders, respectively, across the days of the week.

Specifically, Figure EC.7a shows that the number of in-store customers increases significantly over the weekend, suggesting a preference for physical shopping on Saturdays and Sundays. This pattern aligns with general consumer behavior, where weekends provide more flexibility for in-store visits. In contrast, Figure EC.7b indicates that online order volumes remain relatively stable throughout the week, except for a noticeable drop on Mondays. This suggests that customers may place fewer online orders at the beginning of the week, possibly due to weekend purchases or shifts in shopping behavior.



(a) Customer Distribution by Day of the Week (b) Online Order Distribution by Day of the Week

Figure EC.7 Distribution of the Orders and Store Customers

Understanding not only the daily but also the hourly distribution of store operations is crucial for effective workforce management. To analyze this, we categorized the day into three periods: morning (before 13:00),

afternoon (13:00-17:00), and evening (after 17:00). The distribution of online orders and in-store customer visits across these periods is shown in Figure EC.8.

As illustrated in the figure, online orders peak in the morning, while the number of store customers remains relatively low during this period. This suggests that store personnel may be underutilized in the morning. Conversely, store customer visits increase significantly in the afternoon and evening, reaching their highest level in the evening. Interestingly, the lowest number of online orders is observed in the afternoon, aligning with a period when in-store visits begin to rise. This suggests that customers who visit the store in person during these hours might be substituting their online orders with physical shopping.

These findings emphasize the need for dynamic resource allocation throughout the day. Adjusting staffing levels and operational focus based on time-dependent demand fluctuations can improve both online and in-store efficiency.

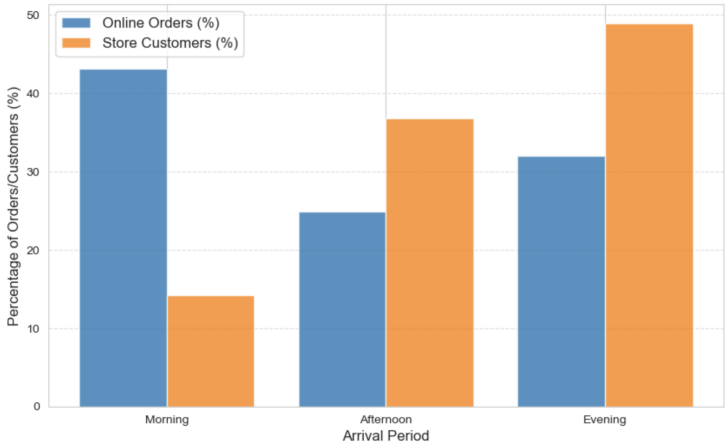
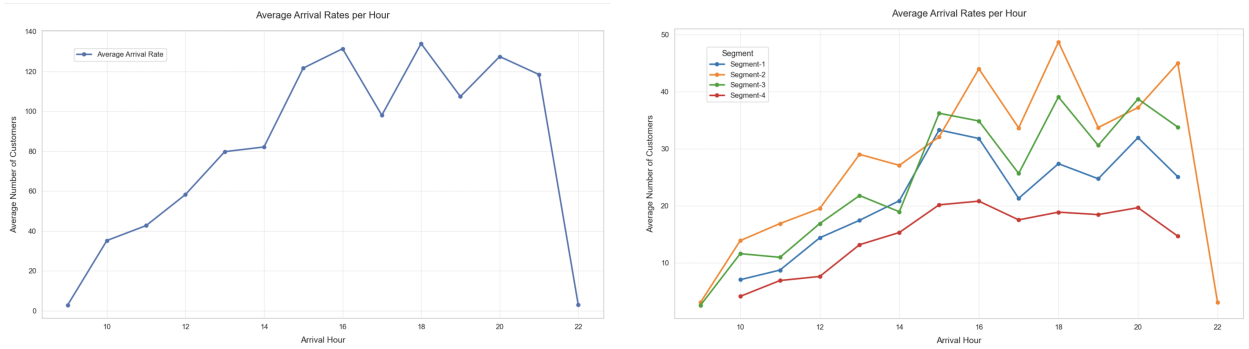


Figure EC.8 Percentage Comparison of Online Orders and Store Customer Arrivals by Time Period

To gain a broader perspective, we extended the analysis by calculating the average hourly arrival rates of customers, as well as segment-based hourly arrival rates, presented in Figure EC.9. As shown in Figure EC.9a the number of customers increases gradually during the morning hours, followed by a sharp rise after 2 PM, reaching its peak in the evening.

Figure EC.9b further illustrates the arrival patterns across different customer segments. It is observed that *Segment-4* maintains a relatively stable arrival rate throughout the day, whereas *Segment-2* displays pronounced peaks, particularly during the late afternoon and evening hours. Since *Segment-4* is intended to serve as a low-cost resource for the model (cheapest per-minute compensation), its stable arrival behavior offers a significant advantage in terms of predictability and operational planning. *Segment-3* also shows moderate fluctuations throughout the day, with a slight increase in the evening hours, while *Segment-1* exhibits a relatively steady but lower arrival rate compared to the other segments. Also, the analysis reveals that *Segment-2* accounts for the highest customer volume, suggesting that, despite its more variable arrival

patterns and high per-minute compensation, it could offer valuable opportunities for task allocation during peak periods.



(a) Hourly Average Customer Arrival Rates (b) Segment-Based Hourly Average Arrival Rates

Figure EC.9 Arrival Rate Analysis

Additionally, we analyzed the distribution of the number of unique product categories in online orders and in-store transactions, as shown in Figure EC.10. This analysis provides valuable insights into customer purchasing behavior across different shopping channels and the complexity of the shopping lists.

As illustrated in the figure, the majority of transactions, both online and in-store, contain fewer than five unique product categories. However, notable differences exist between the two shopping channels. The orange bars, representing online orders, indicate that online shoppers tend to purchase a smaller variety of product categories per transaction. The distribution decreases consistently as the number of product categories increases, suggesting that online purchases are often more focused and specific.

In contrast, the blue bars, representing store transactions, reveal a more varied purchasing pattern. While a large portion of store transactions still falls within the lower category ranges, there is a greater proportion of transactions with a broader selection of product categories compared to online orders. This suggests that in-store shoppers are more likely to engage in exploratory purchasing, potentially adding unplanned items from different categories to their baskets.

Finally, service level is a key performance metric in store operations. In this context, it is defined as the time difference between order placement and delivery, reflecting the store's ability to fulfill customer orders efficiently.

From the analyzed data, the average order delivery time is approximately 4.3 hours. However, as illustrated in FigureEC.11, the distribution of delivery times highlights key operational considerations. A significant portion of orders (35%) are fulfilled within 2 hours or less, indicating a high volume of urgent or same-day deliveries. This trend suggests that a considerable number of customers expect rapid fulfillment. Conversely, 20% of orders are placed the evening before and scheduled for fulfillment the next day.

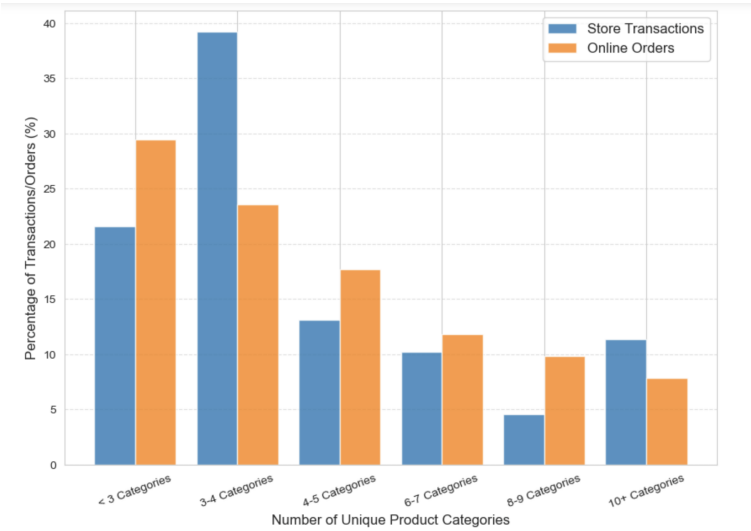


Figure EC.10 Distribution of Unique Product Categories in Online Orders and Store Transactions

It is important to note that delivery times in this system are user-selected, but subject to certain constraints imposed by store operations. The system allows customers to choose delivery slots, with the earliest available option typically set to 1-2 hours after the order is placed, depending on store and online store workload. Customers cannot select an earlier slot, but they can opt for later deliveries based on availability. Additionally, orders placed after 9:00 PM are automatically scheduled for the following day.

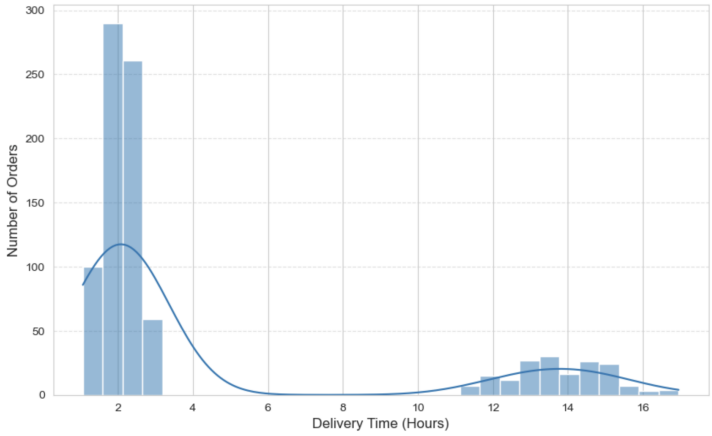


Figure EC.11 Distribution of Order Delivery Times

The analysis of the real omnichannel data reveals that workload distribution and channel priorities dynamically fluctuate over time. Therefore, an efficient assignment model must account for these temporal variations and store-specific dynamics. So, the system is designed to optimize assignments by incorporating online order workload, in-store transactions, delivery time constraints, and expected costs, ensuring a more adaptive and efficient operational framework.

EC.9 Store Layout

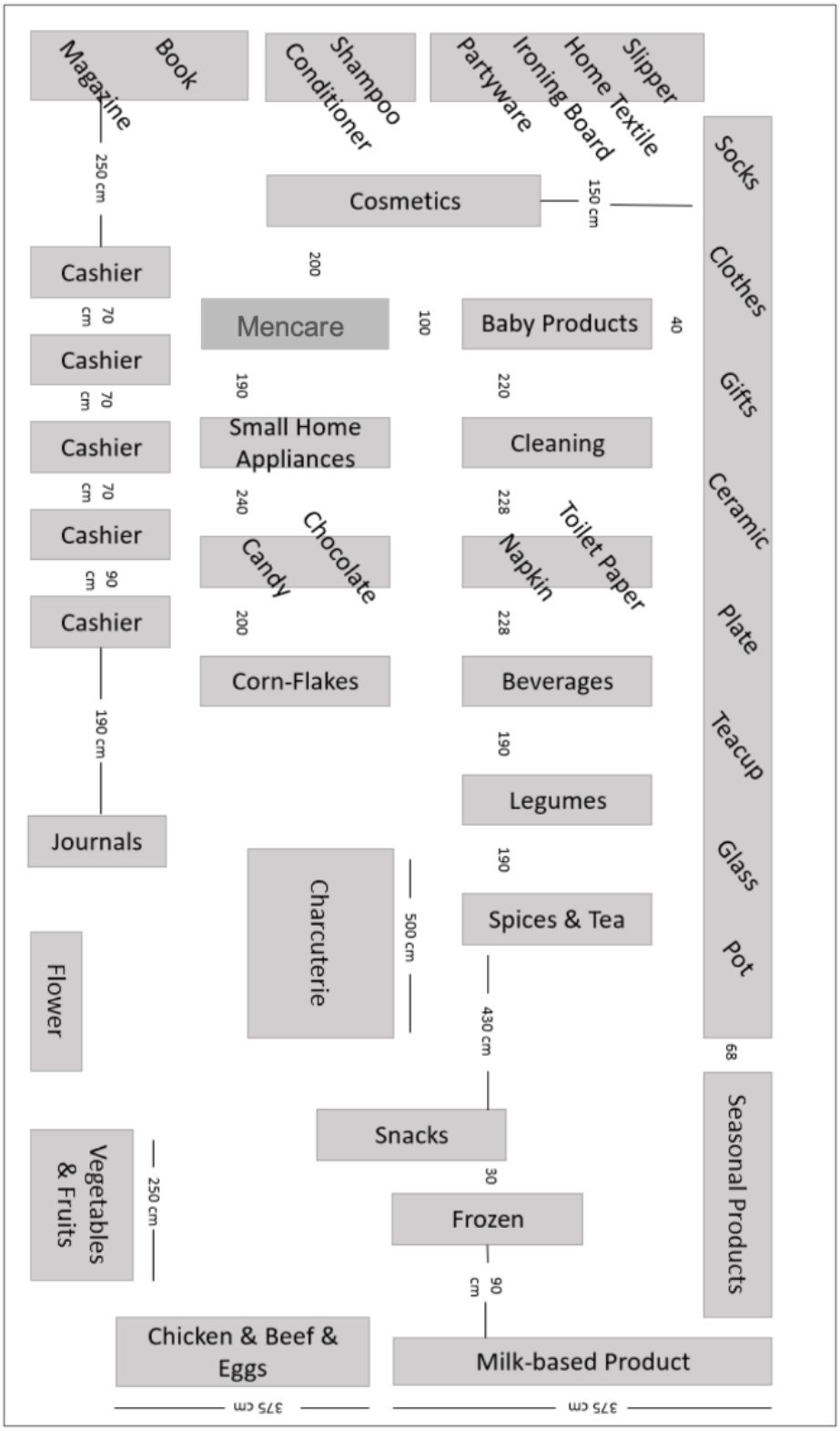


Figure EC.12 Layout of the Supermarket